RM0483
Reference manual

SR5E1x 32-bit Arm® Cortex®-M7 architecture microcontroller for electrical vehicle applications

Overview

This document defines the functionality of the SR5E1x microcontroller for use by software and hardware developers. The SR5E1x is built on Arm® architecture technology and integrates technologies that are important for today's and tomorrow's electrical vehicle applications.

SR5E1x applications include, amongst possible others:

- On-board charger
- DC/DC converters
- Traction inverter
- Advanced control motor

The information in this book is subject to change without notice, as described in the disclaimer. As with any technical documentation, it is the reader's responsibility to be sure he or she is using the most recent version of the documentation.

To locate any published errata or updates for this document, visit the ST Web site at http://www.st.com.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the SR5E1x device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Arm® architecture.

Contents

1 Preface 46

1.1 Document organization 46

1.2 Register conventions 46

1.3 Acronyms and abbreviations 47

1.4 Reference documents 47

2 Introduction 48

2.1 Overall architecture 48

2.1.1 Cortex®-M7 49

2.1.2 Cortex®-M0+ 49

2.1.3 Memory hierarchy 49

2.2 Features 50

2.3 Features list 52

2.4 Block diagram 54

3 System and memory overview 57

3.1 System architecture 57

3.1.1 TCM buses 58

3.1.2 CPU buses 58

3.1.3 Bus master peripherals 59

3.1.4 TCM memories 59

3.2 Memory organization 59

3.3 Embedded SRAM 76

3.4 Flash memory overview 77

4 Signal description 78

4.1 Production packages 78

4.2 Package pinouts and ballouts 78

5 Device configuration 79

5.1 Core modules 79

5.2 System modules 79

5.2.1 Interconnect / bus matrix 79

5.2.2 Hardware semaphore (HSEM2) configuration 85
5.2.3 System memory protection unit (SMPU) configuration 85
5.2.4 Nested vectored interrupt controller configuration 86
5.2.5 EXTI configuration 96
5.2.6 DMA controller configuration 98
5.2.7 DMA request multiplexer (DMAMUX) configuration 99
5.2.8 GPIO configuration 103
5.3 Memories and memory interfaces 103
5.3.1 Non-volatile memory platform controller (NVMPC) configuration 103
5.3.2 Platform RAM controller (PRAMC) 103
5.4 Analog modules 104
5.4.1 Temperature sensor configuration 104
5.4.2 ADCx configuration 104
5.4.3 SDADC configuration 107
5.4.4 DAC configuration 112
5.4.5 Comparator configuration 112
5.5 Timers 112
5.5.1 HRTIMx configuration 112
5.5.2 TIMx timer overview 118
5.5.3 TIM1/TIM8 configuration 120
5.5.4 TIM2/TIM3/TIM4/TIM5 configuration 124
5.5.5 TIM15/TIM16 configuration 127
5.5.6 Timestamp timer configuration 129
5.5.7 Independent watchdog (IWDG) 129
5.5.8 System window watchdog (WWDG) 129
5.5.9 RTC configuration 129
5.6 Communication interfaces 130
5.6.1 CAN subsystem configuration 130
5.6.2 SPI configuration 130
5.6.3 UART configuration 130
5.6.4 I2C configuration 130
5.7 Reset and boot modules 130
5.7.1 Boot assist firmware (BAF) configuration 130
5.7.2 System status and configuration module (SSCM) configuration 131
5.8 Safety modules 131
5.8.1 Cyclic redundancy check (CRC) 131

5.8.2 Memory error management unit (MEMU2) . . . 132
5.8.3 Indirect memory access (IMA) configuration . . . 134
5.8.4 Fault collection & control unit (FCCU) configuration . . . 136
5.8.5 CEM configuration . . . 154
5.8.6 REG_PROT configuration . . . 154
5.8.7 STCU configuration . . . 159
5.8.8 ECC . . . 162

6 Reset and boot . . . 164
6.1 Introduction . . . 164
6.1.1 TEST flash memory block . . . 164
6.1.2 UTEST flash memory block . . . 164
6.1.3 Boot assist flash . . . 164
6.2 Modules used in reset sequence . . . 164
6.2.1 Power management controller . . . 165
6.2.2 Reset and clock control module (RCC) . . . 165
6.2.3 System status and configuration module . . . 166
6.2.4 Self-test control unit . . . 166
6.3 Reset sequence . . . 166
6.3.1 Power-on and reset generation module . . . 166
6.3.2 Power-up phase: power stabilization . . . 169
6.3.3 PHASE0 phase: analog supply initial configuration . . . 170
6.3.4 PHASE1[DEST] phase: temporization setup . . . 170
6.3.5 PHASE2[DEST] phase: flash initial configuration . . . 170
6.3.6 PHASE3[DEST] phase: device configuration . . . 171
6.3.7 IDLE[DEST] phase: self-test execution . . . 172
6.3.8 PHASE1[FUNC] phase: temporization setup . . . 172
6.3.9 PHASE2[FUNC] phase: flash initial configuration . . . 173
6.3.10 PHASE3[FUNC] phase: device configuration monitoring . . . 173
6.3.11 IDLE[FUNC] phase . . . 174
6.3.12 System start-up . . . 174
6.3.13 Waking-up the other CPU . . . 177
6.3.14 BAF serial bootloader . . . 177
6.4 RESETN pin functionality . . . 177
6.5 Reset source mapping . . . 180
6.5.1 Destructive reset sources . . . 181
6.5.2 Functional reset sources . . . 181

6.5.3 Functional reset escalation . . . 182
6.5.4 Destructive reset escalation . . . 183
6.6 RAM preservation upon reset . . . 183

7 Device configuration format (DCF) records . . . 184
7.1 Introduction . . . 184
7.2 DCF clients . . . 184
7.3 DCF records . . . 185
7.3.1 UTEST DCF records . . . 188
7.4 DCF client table . . . 188
7.4.1 DCF client list . . . 189
7.4.2 Miscellaneous DCF records . . . 198
7.4.3 RCC DCF records . . . 203

8 Power management . . . 208
8.1 Overview . . . 208
8.1.1 Power management framework . . . 208
8.1.2 Power management supply description . . . 209
8.1.3 Power management controller overview . . . 210
8.2 Flash power requirements . . . 212
8.3 Device trimming . . . 213
8.4 Supply monitoring (POR and LVDs) . . . 213
8.4.1 Power-on reset (POR) . . . 213
8.4.2 Behavior of device LVD / HVD . . . 213
8.4.3 Voltage detections (MVDs, LVDs, HVDs, UVDs) . . . 214
8.5 Power sequence . . . 218
8.5.1 Power-up sequence . . . 218
8.5.2 Power-down sequence . . . 223
8.5.3 Brown-out management . . . 223
8.5.4 Low voltage requirement during crank . . . 224

9 Security . . . 225
9.1 Basic security . . . 225
9.2 Advanced security . . . 225
9.3 Detailed security information . . . 226

10 Debug support (DBG) 227
10.1 Debug and trace architecture 227
10.2 Debug infrastructure features 227
10.3 Debug access port functional description 228
10.3.1 Serial-wire and JTAG debug port (SWJ-DP) 228
10.3.2 Access ports 228
10.4 NIC ports related to debug 228
10.5 Global timestamp generator (TSG) 228
10.6 Cross trigger interfaces (CTI) and matrix (CTM) 229
10.6.1 Embedded trace FIFO (ETF) 231
10.7 Trace port interface unit (TPIU) 231
10.8 Cortex-M7 debug functional description 231
10.9 Debug configuration in lockstep 231
10.10 Additional requirements 232
10.10.1 Capability of the debugger host to connect under system reset 232
10.10.2 Capability to control debug features of cores and IPs 233
10.10.3 Debug protection 233
10.11 DBGMCU module 233
10.12 DBGMCU register summary 233
10.13 DBGMCU register descriptions 235

11 Reset and clock control (RCC) 253
11.1 Reset 253
11.2 Clocks 253
11.2.1 External oscillator (XOSC) 256
11.2.2 Internal RC oscillator (IRCOSC) 257
11.2.3 PLL 262
11.2.4 Low speed internal RC oscillator (LSIRCOSC) 263
11.2.5 System clock (SYSCLK) selection 263
11.2.6 System clock switching to IRCOSC 264
11.2.7 ADC clock 264
11.2.8 RTC clock 264
11.2.9 LSICLK clock branch 265
11.2.10 Timer clock 265
11.2.11 Watchdog clock 265

11.2.12 Clock-out capability . . . 265
11.2.13 Internal/external clock measurement with TIM5/TIM15/TIM16 . . . 265
11.2.14 Peripheral clock enable register (RCC_AHBxENR, RCC_APBxENR) . . . 267
11.3 Clock monitoring . . . 267
11.3.1 CMU configuration . . . 267
11.3.2 PLL0 monitor . . . 270
11.3.3 External oscillator (XOSC) monitor . . . 270
11.3.4 Internal RC oscillator (IRCOSC) monitor . . . 270
11.3.5 System clock monitors . . . 270
11.3.6 Progressive system clock switching . . . 270
11.4 Low-power modes . . . 273
11.4.1 Peripheral clock gating . . . 273
11.4.2 Core sleep mode (CSleep) . . . 274
11.5 RCC registers . . . 275
11.6 RCC register descriptions . . . 277

12 Dual PLL digital interface (PLLDIG) . . . 328
12.1 Introduction . . . 328
12.2 Block diagram . . . 328
12.3 Features . . . 329
12.4 Modes of operation . . . 329
12.4.1 Normal mode with reference, PLL0 or both PLLs enabled . . . 329
12.5 Memory map and register definition . . . 330
12.5.1 Memory map . . . 330
12.5.2 Register descriptions . . . 330
12.5.3 Register classification for safety requirements . . . 339
12.6 Functional description . . . 339
12.6.1 Input clock frequency . . . 340
12.6.2 Clock configuration . . . 340
12.6.3 Frequency modulation . . . 341
12.7 Initialization information . . . 343

13 Clock monitor unit (CMU) . . . 344
13.1 Introduction . . . 344
13.1.1 Main features . . . 344

13.2 Block diagram 344
13.3 Signals 345
13.4 Memory map and register definition 345
13.4.1 Register descriptions 346
13.5 Functional description 351
13.5.1 Frequency meter 351
13.5.2 CLKMN0_RMT supervisor 352
13.5.3 CLKMN1 supervisor 352

14 Power management controller digital interface (PMC_Dig) 353
14.1 Introduction 353
14.2 Main features 355
14.2.1 Standard features 355
14.3 IPS bus interface 355
14.4 Memory map/register definition 355
14.5 Registers description 358
14.5.1 Event pending register (EPR_LV0) 358
14.5.2 Reset event enable register (REE_LV0) 360
14.5.3 Reset event select register (RES_LV0) 361
14.5.4 Interrupt enable (IE_LV0) 363
14.5.5 FCCU event enable register (FEE_LV0) 365
14.5.6 Event pending register (EPR_LV1) 366
14.5.7 Reset event enable register (REE_LV1) 367
14.5.8 Reset event select register (RES_LV1) 368
14.5.9 Interrupt enable (IE_LV1) 369
14.5.10 FCCU event enable register (FEE_LV1) 370
14.5.11 Event pending register (EPR_HV0) 371
14.5.12 Reset event enable register (REE_HV0) 373
14.5.13 Reset event select register (RES_HV0) 376
14.5.14 Interrupt enable (IE_HV0) 378
14.5.15 FCCU event enable register (FEE_HV0) 380
14.5.16 Supply gauge status register (GR_S) 382
14.5.17 Pending gauge status register (GR_P) 387
14.5.18 Global interrupt enable register (IE_G) 389
14.5.19 HPREG SMPS SELECT STATUS (HPREG_SMPS_SEL_S) 391
14.5.20 Event pending register (EPR_TD) 391

14.5.21 Reset event enable register (REE_TD) . . . 393
14.5.22 Reset event select register (RES_TD) . . . 394
14.5.23 Temperature sensor configuration register (CTL_TD) . . . 395
14.5.24 Temperature sensor FCCU event enable register (FEE_TD) . . . 396
14.5.25 BIST mask status register (BIST_MASK_STATUS) . . . 397
14.5.26 User BIST flags phase1 register (BIST_FLAGS_PHASE1) . . . 398
14.5.27 User BIST flags phase2 register (BIST_FLAGS_PHASE2) . . . 399
14.5.28 User BIST control register (BIST_CTRL) . . . 400
14.5.29 User BIST Time1 and Time0 register (BIST_TIME10) . . . 401
14.5.30 User BIST Time3 and Time2 register (BIST_TIME32) . . . 402
14.5.31 User BIST Time6 and Time5 register (BIST_TIME65) . . . 403
14.5.32 User BIST VD under test monitor register (BIST_DEBUG) . . . 403
14.6 Temperature Sensor Interface Logic . . . 404
14.7 Power on reset (RCC phase gates) . . . 404
14.8 Flash interface (DCF client usage) . . . 405
14.9 Voltage monitor BIST programming . . . 406

15 Platform RAM controller AXI (PRAMC_AXI) . . . 408
15.1 Introduction . . . 408
15.2 Features . . . 408
15.3 Memory map and register description . . . 410
15.3.1 Memory map . . . 410
15.3.2 Register definitions . . . 410
15.4 Functional description . . . 412
15.4.1 Read/write . . . 412
15.5 Initialization/application information . . . 415
15.6 Safety considerations . . . 416
15.6.1 Data Protection . . . 416
15.6.2 MEMU2 interface . . . 418
15.6.3 Memory feedback monitor . . . 419
15.7 Exclusive monitor . . . 419
15.7.1 Support for Exclusive monitor behavior . . . 419

16 Nonvolatile memory platform controller (NVMPC) . . . 421
16.1 Introduction . . . 421
16.1.1 Overview . . . 421

16.1.2 Features . . . 421
16.2 Block interface . . . 422
16.3 Memory map and register definitions . . . 423
16.3.1 Memory map . . . 423
16.3.2 NVM controller registers . . . 423
16.4 AXI4 global/read data/read address interface . . . 430
16.4.1 Operation . . . 430
16.5 Nonvolatile-memory read interface . . . 430
16.5.1 Operation . . . 430
16.6 AXI4 write data/write address/write response interface . . . 432
16.6.1 Operation . . . 433
16.7 Nonvolatile-memory write interface . . . 433
16.7.1 Operation . . . 433
16.8 Functional description . . . 433
16.8.1 AXI4 read slave controller . . . 434
16.8.2 AXI4 write slave controller . . . 438
16.9 Safety . . . 438
16.9.1 End2End ECC . . . 438
16.9.2 Error management . . . 438
16.10 Security . . . 440
16.10.1 Read NVM access protection . . . 440
16.10.2 Data flash management . . . 440
16.11 Over the air (OTA-X1) support . . . 441
16.11.1 Overview . . . 441
16.11.2 Architecture . . . 441

17 Embedded flash memory . . . 445
17.1 Introduction . . . 445
17.1.1 Overview . . . 445
17.1.2 Features . . . 448
17.1.3 Modes of operation . . . 448
17.2 Flash memory map and description . . . 449
17.2.1 Flash array memory map . . . 449
17.3 Register memory maps and descriptions . . . 453
17.3.1 Register memory maps . . . 454
17.3.2 User register descriptions . . . 455

17.4 Functional description . . . 513
17.4.1 Reset . . . 513
17.4.2 Read mode . . . 514
17.4.3 Modify mode . . . 517
17.4.4 Alternate program and erase interface . . . 524
17.4.5 User Test mode . . . 526
17.4.6 Protection strategy . . . 531

18 Boot assist flash (BAF) . . . 533
18.1 Introduction . . . 533
18.2 Memory map . . . 533
18.2.1 BAF image header . . . 533
18.3 Functional description . . . 534
18.3.1 Boot modes . . . 535
18.3.2 Device initialization . . . 535
18.3.3 Flow of control . . . 535
18.3.4 Optionally wait for HSM . . . 536
18.3.5 Initialization of BAF DCF clients . . . 537
18.3.6 Production disable activation protocol and implementation . . . 541
18.3.7 TDM diary operation . . . 542
18.3.8 Optionally perform a serial boot . . . 542
18.3.9 Serial boot configuration . . . 546
18.4 Resources accessed by BAF code execution . . . 549

19 System status and configuration module (SSCM) . . . 550
19.1 Introduction . . . 550
19.1.1 Overview . . . 550
19.1.2 Features . . . 550
19.1.3 Modes of operation . . . 550
19.2 Memory map and registers definition . . . 550
19.2.1 Registers descriptions . . . 551
19.3 Functional description . . . 556
19.3.1 DCF mechanism . . . 556
19.3.2 ECC error monitoring . . . 558
19.3.3 Life cycle . . . 559
19.4 Initialization and application information . . . 560

19.4.1 Reset . . . 560
19.5 Additional safety measures . . . 560
19.5.1 Spurious reset protection . . . 560

20 General-purpose I/Os (GPIO) . . . 561
20.1 Introduction . . . 561
20.2 GPIO main features . . . 561
20.3 GPIO functional description . . . 561
20.3.1 General-purpose I/O (GPIO) . . . 563
20.3.2 I/O pin alternate function multiplexer and mapping . . . 563
20.3.3 I/O port control registers . . . 564
20.3.4 I/O port data registers . . . 564
20.3.5 I/O data bitwise handling . . . 564
20.3.6 GPIO locking mechanism . . . 565
20.3.7 I/O alternate function input/output . . . 565
20.3.8 External interrupt/wakeup lines . . . 565
20.3.9 Input configuration . . . 566
20.3.10 Output configuration . . . 566
20.3.11 Alternate function configuration . . . 567
20.3.12 Analog configuration . . . 568
20.3.13 Safe Mode Configuration (SMC) of GPIOs . . . 568
20.4 GPIO registers . . . 569
20.4.1 GPIO memory map . . . 569
20.4.2 Mode register (MODER) (x =A to I) . . . 569
20.4.3 Output type register (OTYPER) (x = A to I) . . . 570
20.4.4 Output speed register (OSPEEDR) (x = A to I) . . . 570
20.4.5 Pull-up/pull-down register (PUPDR) (x = A to I) . . . 571
20.4.6 Input data register (IDR) (x = A to I) . . . 571
20.4.7 Output data register (ODR) (x = A to I) . . . 572
20.4.8 Bit set/reset register (BSRR) (x = A to I) . . . 572
20.4.9 Configuration lock register (LCKR) (x = A to I) . . . 572

20.4.10 Alternate function low register (AFRL) (x = A to I) . . . 573
20.4.11 Alternate function high register (AFRH) (x = A to I) . . . 574
20.4.12 Port Hysteresis Register (IHYSTR) (x = A to I) . . . 575
20.4.13 Port Input Buffer Enable Register (TRIGENR) (x = A to I) . . . 575
20.4.14 Safe IO Selection Register (SAFESELR) (x = A to I) . . . 576
20.4.15 Safe IO Output Value Register (SAFEVALR) (x = A to I) . . . 576

21 System configuration controller (SYSCFG) . . . 578
21.1 SYSCFG main features . . . 578
21.2 SYSCFG register summary . . . 578
21.3 SYSCFG register description . . . 579

22 System memory protection unit (SMPU) . . . 597
22.1 Overview . . . 597
22.2 Block diagram . . . 597
22.3 Features . . . 599
22.4 Memory map and register definition . . . 599
22.4.1 Memory map . . . 599
22.4.2 Register descriptions . . . 600
22.5 Functional description . . . 608
22.5.1 Access evaluation macro . . . 608
22.5.2 Putting it all together and error terminations . . . 610
22.6 Initialization information . . . 610
22.7 Application information . . . 610

23 Direct memory access controller (DMA) . . . 613
23.1 DMA introduction . . . 613
23.2 DMA main features . . . 613
23.3 DMA functional description . . . 615
23.3.1 DMA block diagram . . . 615
23.3.2 DMA internal signals . . . 615
23.3.3 DMA overview . . . 615

23.3.4 DMA transactions . . . 616
23.3.5 DMA request mapping . . . 616
23.3.6 Arbiter . . . 617
23.3.7 DMA streams . . . 617
23.3.8 Source, destination and transfer modes . . . 617
23.3.9 Pointer incrementation . . . 621
23.3.10 Circular mode . . . 622
23.3.11 Double-buffer mode . . . 622
23.3.12 Programmable data width, packing/unpacking, endianness . . . 623
23.3.13 Single and burst transfers . . . 625
23.3.14 FIFO . . . 626
23.3.15 DMA transfer completion . . . 628
23.3.16 DMA transfer suspension . . . 629
23.3.17 Flow controller . . . 630
23.3.18 Summary of the possible DMA configurations . . . 631
23.3.19 Stream configuration procedure . . . 631
23.3.20 Error management . . . 632
23.4 DMA interrupts . . . 633
23.5 DMA registers . . . 634
23.5.1 DMA memory map . . . 634
23.5.2 DMA low interrupt status register (DMA_LISR) . . . 634
23.5.3 DMA high interrupt status register (DMA_HISR) . . . 635
23.5.4 DMA low interrupt flag clear register (DMA_LIFCR) . . . 636
23.5.5 DMA high interrupt flag clear register (DMA_HIFCR) . . . 636
23.5.6 DMA stream x configuration register (DMA_SxCR) . . . 637
23.5.7 DMA stream x number of data register (DMA_SxNDTR) . . . 640
23.5.8 DMA stream x peripheral address register (DMA_SxPAR) . . . 641
23.5.9 DMA stream x memory 0 address register (DMA_SxM0AR) . . . 641
23.5.10 DMA stream x memory 1 address register (DMA_SxM1AR) . . . 642
23.5.11 DMA stream x FIFO control register (DMA_SxFCR) . . . 642

24 DMA request multiplexer (DMAMUX) . . . 644
24.1 Introduction . . . 644
24.2 DMAMUX main features . . . 645
24.3 DMAMUX functional description . . . 646

24.3.1 DMAMUX block diagram . . . 646
24.3.2 DMAMUX signals . . . 647
24.3.3 DMAMUX channels . . . 647
24.3.4 DMAMUX request line multiplexer . . . 647
24.3.5 DMAMUX request generator . . . 650
24.4 DMAMUX interrupts . . . 651
24.5 DMAMUX registers . . . 652
24.5.1 DMAMUX memory map . . . 652
24.5.2 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR) . . . 652
24.5.3 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR) . . . 653
24.5.4 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR) . . . 654
24.5.5 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR) . . . 654
24.5.6 DMAMUX request generator interrupt status register (DMAMUX_RGSR) . . . 655
24.5.7 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR) . . . 655

25 Nested vectored interrupt controller (NVIC) . . . 656
25.1 NVIC main features . . . 656
25.2 SysTick calibration value register . . . 656

26 Extended interrupt and event controller (EXTI) . . . 657
26.1 EXTI main features . . . 657
26.2 EXTI block diagram . . . 657
26.2.1 EXTI connections between peripherals and CPU . . . 658
26.3 EXTI functional description . . . 658
26.3.1 EXTI configurable event input CPU wakeup . . . 659
26.3.2 EXTI configurable event input Any wakeup . . . 660
26.3.3 EXTI direct event input CPU wakeup . . . 661
26.3.4 EXTI direct event input Any wakeup . . . 663
26.4 EXTI event input mapping . . . 663
26.5 EXTI functional behavior . . . 664
26.5.1 EXTI CPU interrupt procedure . . . 665
26.5.2 EXTI CPU event procedure . . . 665

26.5.3 EXTI software interrupt/event trigger procedure 665
26.6 EXTI registers 666
26.6.1 EXTI memory map 666
26.6.2 Rising trigger selection register (RTSR1) 666
26.6.3 Falling trigger selection register (FTSR1) 667
26.6.4 Software interrupt event register (SWIER1) 668
26.6.5 Rising trigger selection register (RTSR2) 668
26.6.6 Falling trigger selection register (FTSR2) 669
26.6.7 Software interrupt event register (SWIER2) 669
26.6.8 Interrupt mask register (CnIMR1) 670
26.6.9 Event mask register (CnEMR1) 671
26.6.10 Pending register (CnPR1) 671
26.6.11 Interrupt mask register (CnIMR2) 672
26.6.12 Event mask register (CnEMR2) 673
26.6.13 Pending register (CnPR2) 673

27 Cyclic redundancy check calculation unit (CRC) 674
27.1 Introduction 674
27.2 CRC main features 674
27.3 CRC functional description 675
27.3.1 CRC block diagram 675
27.3.2 CRC internal signals 675
27.3.3 CRC operation 675
27.4 CRC registers 677
27.4.1 CRC memory map 677
27.4.2 CRC data register (CRC_DR) 677
27.4.3 CRC independent data register (CRC_IDR) 677
27.4.4 CRC control register (CRC_CR) 678
27.4.5 CRC initial value (CRC_INIT) 679
27.4.6 CRC polynomial (CRC_POL) 679

28 CORDIC coprocessor (CORDIC) 680
28.1 CORDIC introduction 680
28.2 CORDIC main features 680
28.3 CORDIC functional description 680
28.3.1 General description 680

28.3.2 CORDIC functions . . . 680
28.3.3 Fixed point representation . . . 687
28.3.4 Scaling factor . . . 687
28.3.5 Precision . . . 688
28.3.6 Zero-overhead mode . . . 691
28.3.7 Polling mode . . . 692
28.3.8 Interrupt mode . . . 693
28.3.9 DMA mode . . . 693
28.4 CORDIC registers . . . 694
28.4.1 CORDIC memory map . . . 694
28.4.2 CORDIC control/status register (CORDIC_CSR) . . . 694
28.4.3 CORDIC argument register (CORDIC_WDATA) . . . 696
28.4.4 CORDIC result register (CORDIC_RDATA) . . . 697

29 Analog-to-digital converters (ADC) . . . 698
29.1 Introduction . . . 698
29.2 ADC main features . . . 699
29.3 ADC implementation . . . 700
29.4 ADC functional description . . . 701
29.4.1 ADC block diagram . . . 701
29.4.2 ADC pins and internal signals . . . 702
29.4.3 ADC clocks . . . 703
29.4.4 Slave AHB interface . . . 704
29.4.5 ADC voltage regulator (ADVREGEN) . . . 704
29.4.6 Single-ended and differential input channels . . . 705
29.4.7 Calibration (ADCAL, ADCALDIF, ADC_CALFACT) . . . 705
29.4.8 ADC on-off control (ADEN, ADDIS, ADRDY) . . . 708
29.4.9 Constraints when writing the ADC control bits . . . 709
29.4.10 Channel selection (SQRx, JSQRx) . . . 710
29.4.11 Channel-wise programmable sampling time (SMPR1, SMPR2) . . . 711
29.4.12 Single conversion mode (CONT = 0) . . . 713
29.4.13 Continuous conversion mode (CONT = 1) . . . 713
29.4.14 Starting conversions (ADSTART, JADSTART) . . . 714
29.4.15 ADC timing . . . 715
29.4.16 Stopping an ongoing conversion (ADSTP, JADSTP) . . . 715
29.4.17 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN) . . . 717

29.4.18 Injected channel management 718
29.4.19 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) 720
29.4.20 Queue of context for injected conversions 721
29.4.21 Programmable resolution (RES) - fast conversion mode 729
29.4.22 End of conversion, end of sampling phase (EOC, JEOC, EOSMP) . . 730
29.4.23 End of conversion sequence (EOS, JEOS) 730
29.4.24 Timing diagrams example (Single/Continuous modes, hardware/software triggers) 731
29.4.25 Data management 733
29.4.26 Dynamic low-power features 739
29.4.27 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx) 744
29.4.28 Oversampler 748
29.4.29 Dual ADC modes 754
29.5 ADC interrupts 769
29.6 ADC registers 770
29.6.1 ADC register memory map (for each ADC) 770
29.6.2 ADC common registers 798

30 Sigma-delta analog-to-digital converter (SDADC) digital interface 803

30.1 Introduction 803
30.2 Overview 803
30.3 Features 805
30.4 Modes of operation 805
30.4.1 Differential input mode 805
30.4.2 Single-ended input mode 805
30.4.3 External modulator mode 806
30.5 External signal description 806
30.5.1 Detailed signal descriptions 806
30.6 Memory map and register descriptions 807
30.6.1 Memory map 807
30.6.2 Registers description 808
30.7 Functional description 824
30.7.1 Differential input mode 824
30.7.2 Single-ended input mode 825
30.7.3 External modulator mode 825

30.7.4 Low consumption mode . . . 825
30.7.5 Analog input multiplexing and bias support . . . 825
30.7.6 Programmable gain and decimation rate . . . 825
30.7.7 SD ADC Internal Filter Selection . . . 826
30.7.8 High-pass filter support . . . 828
30.7.9 Data conversion . . . 828
30.7.10 Hardware triggering . . . 829
30.7.11 Watchdog . . . 830
30.7.12 Interrupt/DMA request support . . . 831
30.7.13 Gain calibration support . . . 831
30.7.14 Offset calibration support . . . 833
30.7.15 Global gating . . . 833
30.7.16 Timestamp . . . 834
30.8 Initialization information . . . 834
30.8.1 Data conversion step . . . 835

31 Digital-to-analog converter (DAC) . . . 836
31.1 Introduction . . . 836
31.2 DAC main features . . . 836
31.3 DAC implementation . . . 837
31.4 DAC functional description . . . 838
31.4.1 DAC block diagram . . . 838
31.4.2 DAC pins and internal signals . . . 839
31.4.3 DAC channel enable . . . 844
31.4.4 DAC data format . . . 844
31.4.5 DAC conversion . . . 846
31.4.6 DAC output voltage . . . 847
31.4.7 DAC trigger selection . . . 847
31.4.8 DMA requests . . . 847
31.4.9 DAC noise generation . . . 848
31.4.10 DAC triangle-wave generation . . . 850
31.4.11 DAC sawtooth wave generation . . . 851
31.4.12 DAC channel modes . . . 852
31.4.13 DAC channel buffer calibration . . . 855
31.4.14 Dual DAC channel conversion modes (if dual channels are available) . . . 856
31.5 DAC low-power modes . . . 861

31.6 DAC interrupts 861
31.7 DAC registers 862
31.7.1 DAC memory map 862
31.7.2 DAC control register (DAC_CR) 863
31.7.3 DAC software trigger register (DAC_SWTRGR) 866
31.7.4 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) 867
31.7.5 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) 868
31.7.6 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) 868
31.7.7 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) 869
31.7.8 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) 869
31.7.9 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) 870
31.7.10 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) 870
31.7.11 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) 871
31.7.12 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) 871
31.7.13 DAC channel1 data output register (DAC_DOR1) 872
31.7.14 DAC channel2 data output register (DAC_DOR2) 872
31.7.15 DAC status register (DAC_SR) 873
31.7.16 DAC calibration control register (DAC_CCR) 874
31.7.17 DAC mode control register (DAC_MCR) 875
31.7.18 DAC channel1 sample and hold sample time register (DAC_SHSR1) 876
31.7.19 DAC channel2 sample and hold sample time register (DAC_SHSR2) 877
31.7.20 DAC sample and hold time register (DAC_SHHR) 877
31.7.21 DAC sample and hold refresh time register (DAC_SHRR) 878
31.7.22 DAC channel1 sawtooth register (DAC_STR1) 879
31.7.23 DAC channel2 sawtooth register (DAC_STR2) 879
31.7.24 DAC sawtooth mode register (DAC_STMODR) 880

32 Comparator (COMP) 882

32.1 COMP introduction 882

32.2 COMP main features . . . 882
32.2.1 COMP block diagram . . . 883
32.2.2 COMP pins and internal signals . . . 883
32.2.3 COMP reset and clocks . . . 884
32.2.4 COMP LOCK mechanism . . . 884
32.2.5 COMP hysteresis . . . 884
32.2.6 COMP output blanking . . . 885
32.3 COMP low-power modes . . . 886
32.4 COMP interrupts . . . 886
32.5 COMP_Dig registers . . . 887
32.5.1 COMP register map . . . 887
32.5.2 Comparator x control and status register (COMP_CxCSR) . . . 887

33 Temperature sensor . . . 889
33.1 Introduction . . . 889
33.1.1 Signal description . . . 889
33.2 Functional description . . . 889
33.2.1 Linear temperature sensor (analog output generation) . . . 889
33.3 Temperature formula . . . 889
33.3.1 Equations for converting TSENS voltages to junction temperature . . . 890
33.3.2 Equations for converting TSENS voltages into constant reference (Digital bandgap voltage) . . . 890

34 High-resolution timer (HRTIM) . . . 892
34.1 Introduction . . . 892
34.2 Main features . . . 892
34.3 Functional description . . . 893
34.3.1 General description . . . 893
34.3.2 HRTIM pins and internal signals . . . 896
34.3.3 Clocks . . . 898
34.3.4 Timer A..F timing units . . . 901
34.3.5 Master timer . . . 923
34.3.6 Up-down counting mode . . . 924
34.3.7 Set/reset event priorities and narrow pulse management . . . 931
34.3.8 External events global conditioning . . . 935
34.3.9 External event filtering in timing units . . . 940

34.3.10 Delayed protection . . . 946
34.3.11 Register preload and update management . . . 952
34.3.12 PWM mode with “greater than” comparison . . . 956
34.3.13 Events propagation within or across multiple timers . . . 957
34.3.14 Output management . . . 962
34.3.15 Burst mode controller . . . 964
34.3.16 Chopper . . . 972
34.3.17 Fault protection . . . 973
34.3.18 Auxiliary outputs . . . 978
34.3.19 Synchronizing the HRTIM with other timers or HRTIM instances . . . 980
34.3.20 ADC triggers . . . 983
34.3.21 DAC triggers . . . 987
34.3.22 Interrupts . . . 991
34.3.23 DMA . . . 993
34.3.24 HRTIM initialization . . . 997
34.3.25 Debug . . . 998
34.4 Application use cases . . . 999
34.4.1 Buck converter . . . 999
34.4.2 Buck converter with synchronous rectification . . . 1000
34.4.3 Multiphase converters . . . 1001
34.4.4 Transition mode power factor correction . . . 1002
34.5 HRTIM registers . . . 1005
34.5.1 HRTIM register memory map . . . 1005
34.5.2 HRTIM master timer control register (HRTIM_MCR) . . . 1010
34.5.3 HRTIM master timer interrupt status register (HRTIM_MISR) . . . 1013
34.5.4 HRTIM master timer interrupt clear register (HRTIM_MICR) . . . 1014
34.5.5 HRTIM master timer DMA interrupt enable register (HRTIM_MDIER) . . . 1015
34.5.6 HRTIM master timer counter register (HRTIM_MCNTR) . . . 1016
34.5.7 HRTIM master timer period register (HRTIM_MPER) . . . 1017
34.5.8 HRTIM master timer repetition register (HRTIM_MREP) . . . 1017
34.5.9 HRTIM master timer compare 1 register (HRTIM_MCMP1R) . . . 1017
34.5.10 HRTIM master timer compare 2 register (HRTIM_MCMP2R) . . . 1018
34.5.11 HRTIM master timer compare 3 register (HRTIM_MCMP3R) . . . 1018
34.5.12 HRTIM master timer compare 4 register (HRTIM_MCMP4R) . . . 1019
34.5.13 HRTIM timer x control register (HRTIM_TIMxCR) (x = A to F) . . . 1019

34.5.14 HRTIM timer x interrupt status register (HRTIM_TIMxISR) (x = A to F) 1023
34.5.15 HRTIM timer x interrupt clear register (HRTIM_TIMxICR) (x = A to F) 1025
34.5.16 HRTIM timer x DMA interrupt enable register (HRTIM_TIMxDIER) (x = A to F) 1027
34.5.17 HRTIM timer x counter register (HRTIM_CNTxR) (x = A to F) 1029
34.5.18 HRTIM timer x period register (HRTIM_PERxR) (x = A to F) 1030
34.5.19 HRTIM timer x repetition register (HRTIM_REPxR) (x = A to F) 1031
34.5.20 HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F) . . 1031
34.5.21 HRTIM timer x compare 1 compound register (HRTIM_CMP1CxR) (x = A to F) 1032
34.5.22 HRTIM timer x compare 2 register (HRTIM_CMP2xR) (x = A to F) . . 1033
34.5.23 HRTIM timer x compare 3 register (HRTIM_CMP3xR) (x = A to F) . . 1033
34.5.24 HRTIM timer x compare 4 register (HRTIM_CMP4xR) (x = A to F) . . 1034
34.5.25 HRTIM timer x capture 1 register (HRTIM_CPT1xR) (x = A to F) . . . 1035
34.5.26 HRTIM timer x capture 2 register (HRTIM_CPT2xR) (x = A to F) 1036
34.5.27 HRTIM timer x deadtime register (HRTIM_DTxR) (x = A to F) 1037
34.5.28 HRTIM timer x output 1 set register (HRTIM_SETx1R) (x = A to F) . 1038
34.5.29 HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F) 1040
34.5.30 HRTIM timer x output 2 set register (HRTIM_SETx2R) (x = A to F) . 1041
34.5.31 HRTIM timer x output 2 reset register (HRTIM_RSTx2R) (x = A to F) 1041
34.5.32 HRTIM timer x external event filtering register 1 (HRTIM_EEFxR1) (x = A to F) 1042
34.5.33 HRTIM timer x external event filtering register 2 (HRTIM_EEFxR2) (x = A to F) 1044
34.5.34 HRTIM timer A reset register (HRTIM_RSTAR) 1045
34.5.35 HRTIM timer B reset register (HRTIM_RSTBR) 1047
34.5.36 HRTIM timer C reset register (HRTIM_RSTCR) 1049
34.5.37 HRTIM timer D reset register (HRTIM_RSTDR) 1051
34.5.38 HRTIM timer E reset register (HRTIM_RSTER) 1053
34.5.39 HRTIM timer F reset register (HRTIM_RSTFR) 1055
34.5.40 HRTIM timer x chopper register (HRTIM_CHPxR) (x = A to F) 1057
34.5.41 HRTIM timer A capture 1 control register (HRTIM_CPT1ACR) 1058
34.5.42 HRTIM timer B capture 1 control register (HRTIM_CPT1BCR) 1058
34.5.43 HRTIM timer C capture 1 control register (HRTIM_CPT1CCR) 1058
34.5.44 HRTIM timer D capture 1 control register (HRTIM_CPT1DCR) 1059
34.5.45 HRTIM timer E capture 1 control register (HRTIM_CPT1ECR) 1059

34.5.46 HRTIM timer F capture 1 control register (HRTIM_CPT1FCR) 1059
34.5.47 HRTIM timer x capture 2 control register (HRTIM_CPT2xCR) (x = A to F) 1060
34.5.48 HRTIM timer x output register (HRTIM_OUTxR) (x = A to F) 1065
34.5.49 HRTIM timer x fault register (HRTIM_FLTxR) (x = A to F) 1068
34.5.50 HRTIM timer x control register 2 (HRTIM_TIMxCR2) (x = A to F) . . . 1069
34.5.51 HRTIM timer x external event filtering register 3 (HRTIM_EEFxR3) (x = A to F) 1071
34.5.52 HRTIM control register 1 (HRTIM_CR1) 1072
34.5.53 HRTIM control register 2 (HRTIM_CR2) 1073
34.5.54 HRTIM interrupt status register (HRTIM_ISR) 1075
34.5.55 HRTIM interrupt clear register (HRTIM_ICR) 1076
34.5.56 HRTIM interrupt enable register (HRTIM_IER) 1077
34.5.57 HRTIM output enable register (HRTIM_OENR) 1078
34.5.58 HRTIM output disable register (HRTIM_ODISR) 1079
34.5.59 HRTIM output disable status register (HRTIM_ODSR) 1080
34.5.60 HRTIM burst mode control register (HRTIM_BMCR) 1081
34.5.61 HRTIM burst mode trigger register (HRTIM_BMTRGR) 1083
34.5.62 HRTIM burst mode compare register (HRTIM_BMCMPR) 1085
34.5.63 HRTIM burst mode period register (HRTIM_BMPER) 1086
34.5.64 HRTIM timer external event control register 1 (HRTIM_EECR1) . . . 1086
34.5.65 HRTIM timer external event control register 2 (HRTIM_EECR2) . . . 1088
34.5.66 HRTIM timer external event control register 3 (HRTIM_EECR3) . . . 1089
34.5.67 HRTIM ADC trigger 1 register (HRTIM_ADC1R) 1090
34.5.68 HRTIM ADC trigger 2 register (HRTIM_ADC2R) 1091
34.5.69 HRTIM ADC trigger 3 register (HRTIM_ADC3R) 1091
34.5.70 HRTIM ADC trigger 4 register (HRTIM_ADC4R) 1093
34.5.71 HRTIM DLL control register (HRTIM_DLLCR) 1095
34.5.72 HRTIM fault input register 1 (HRTIM_FLTINR1) 1096
34.5.73 HRTIM fault input register 2 (HRTIM_FLTINR2) 1097
34.5.74 HRTIM burst DMA master timer update register (HRTIM_BDMUPR) 1099
34.5.75 HRTIM burst DMA timer x update register (HRTIM_BDTxUPR) (x = A to F) 1100
34.5.76 HRTIM burst DMA data register (HRTIM_BDMADR) 1102
34.5.77 HRTIM ADC extended trigger register (HRTIM_ADCER) 1102
34.5.78 HRTIM ADC trigger update register (HRTIM_ADCUR) 1104
34.5.79 HRTIM ADC post scaler register 1 (HRTIM_ADCPS1) 1105

34.5.80 HRTIM ADC post scaler register 2 (HRTIM_ADCPS2) 1106
34.5.81 HRTIM fault input register 3 (HRTIM_FLTINR3) 1107
34.5.82 HRTIM fault input register 4 (HRTIM_FLTINR4) 1108

35 Advanced-control timers (TIM1/TIM8) 1110

35.1 TIM1/TIM8 introduction 1110
35.2 TIM1/TIM8 main features 1110
35.3 TIM1/TIM8 functional description 1111
35.3.1 Block diagram 1111
35.3.2 TIM1/TIM8 pins and internal signals 1112
35.3.3 Time-base unit 1114
35.3.4 Counter modes 1116
35.3.5 Repetition counter 1128
35.3.6 External trigger input 1129
35.3.7 Clock selection 1130
35.3.8 Capture/compare channels 1134
35.3.9 Input capture mode 1136
35.3.10 PWM input mode 1137
35.3.11 Forced output mode 1138
35.3.12 Output compare mode 1139
35.3.13 PWM mode 1140
35.3.14 Asymmetric PWM mode 1148
35.3.15 Combined PWM mode 1149
35.3.16 Combined 3-phase PWM mode 1150
35.3.17 Complementary outputs and dead-time insertion 1151
35.3.18 Using the break function 1154
35.3.19 Bidirectional break inputs 1160
35.3.20 Clearing the tim_ocxref signal on an external event 1161
35.3.21 6-step PWM generation 1163
35.3.22 One-pulse mode 1164
35.3.23 Retriggerable one pulse mode 1165
35.3.24 Pulse on compare mode 1166
35.3.25 Encoder interface mode 1168
35.3.26 Direction bit output 1186
35.3.27 UIF bit remapping 1187
35.3.28 Timer input XOR function 1187
35.3.29 Interfacing with Hall sensors 1187

35.3.30 Timer synchronization 1189
35.3.31 ADC synchronization 1193
35.3.32 DMA burst mode 1194
35.3.33 TIM1/TIM8 DMA requests 1195
35.3.34 Debug mode 1195
35.4 TIM1/TIM8 low-power modes 1195
35.5 TIM1/TIM8 interrupts 1195
35.6 TIM1/TIM8 registers 1197
35.6.1 TIM1/TIM8 memory map 1197
35.6.2 TIMx control register 1 (TIMx_CR1)(x = 1, 8) 1198
35.6.3 TIMx control register 2 (TIMx_CR2)(x = 1, 8) 1199
35.6.4 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8) 1202
35.6.5 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8) 1206
35.6.6 TIMx status register (TIMx_SR)(x = 1, 8) 1207
35.6.7 TIMx event generation register (TIMx_EGR)(x = 1, 8) 1210
35.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8) 1211
35.6.9 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8) 1213
35.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8) 1215
35.6.11 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8) 1216
35.6.12 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8) 1219
35.6.13 TIMx counter (TIMx_CNT)(x = 1, 8) 1223
35.6.14 TIMx prescaler (TIMx_PSC)(x = 1, 8) 1223
35.6.15 TIMx auto-reload register (TIMx_ARR)(x = 1, 8) 1224
35.6.16 TIMx repetition counter register (TIMx_RCR)(x = 1, 8) 1224
35.6.17 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8) 1225
35.6.18 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8) 1225
35.6.19 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8) 1226
35.6.20 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8) 1227
35.6.21 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8) 1228
35.6.22 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8) 1232
35.6.23 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8) 1233
35.6.24 TIMx capture/compare mode register 3 (TIMx_CCMR3)(x = 1, 8) 1234

35.6.25 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8) 1234
35.6.26 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8) 1235
35.6.27 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8) 1236
35.6.28 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8) 1237
35.6.29 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8) 1240
35.6.30 TIMx DMA control register (TIMx_DCR)(x = 1, 8) 1243
35.6.31 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8) 1244

36 General-purpose timers (TIM2/TIM3/TIM4/TIM5) 1245

36.1 TIM2/TIM3/TIM4/TIM5 introduction 1245
36.2 TIM2/TIM3/TIM4/TIM5 main features 1245
36.3 TIM2/TIM3/TIM4/TIM5 implementation 1246
36.4 TIM2/TIM3/TIM4/TIM5 functional description 1247
36.4.1 Block diagram 1247
36.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals 1248
36.4.3 Time-base unit 1250
36.4.4 Counter modes 1252
36.4.5 Clock selection 1263
36.4.6 Capture/compare channels 1267
36.4.7 Input capture mode 1269
36.4.8 PWM input mode 1270
36.4.9 Forced output mode 1271
36.4.10 Output compare mode 1271
36.4.11 PWM mode 1273
36.4.12 Asymmetric PWM mode 1281
36.4.13 Combined PWM mode 1282
36.4.14 Clearing the tim_ocxref signal on an external event 1283
36.4.15 One-pulse mode 1285
36.4.16 Retriggerable one pulse mode 1286
36.4.17 Pulse on compare mode 1287
36.4.18 Encoder interface mode 1289
36.4.19 Direction bit output 1307
36.4.20 UIF bit remapping 1308
36.4.21 Timer input XOR function 1308
36.4.22 Timers and external trigger synchronization 1308
36.4.23 Timer synchronization 1312
36.4.24 DMA burst mode 1317

36.4.25 TIM2/TIM3/TIM4/TIM5 DMA requests 1318
36.4.26 Debug mode 1319
36.4.27 TIM2/TIM3/TIM4/TIM5 low-power modes 1319
36.4.28 TIM2/TIM3/TIM4/TIM5 interrupts 1319
36.5 TIM2/TIM3/TIM4/TIM5 registers 1320
36.5.1 TIM2/TIM3/TIM4/TIM5 memory map 1320
36.5.2 TIMx control register 1 (TIMx_CR1)(x = 2 to 5) 1321
36.5.3 TIMx control register 2 (TIMx_CR2)(x = 2 to 5) 1323
36.5.4 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5) 1324
36.5.5 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5) 1328
36.5.6 TIMx status register (TIMx_SR)(x = 2 to 5) 1329
36.5.7 TIMx event generation register (TIMx_EGR)(x = 2 to 5) 1331
36.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5) 1332
36.5.9 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5) 1334
36.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5) 1336
36.5.11 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5) 1337
36.5.12 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5) . . . 1338
36.5.13 TIMx counter (TIMx_CNT)(x = 3, 4) 1340
36.5.14 TIMx counter (TIMx_CNT)(x = 2, 5) 1340
36.5.15 TIMx prescaler (TIMx_PSC)(x = 2 to 5) 1341
36.5.16 TIMx auto-reload register (TIMx_ARR)(x = 3, 4) 1341
36.5.17 TIMx auto-reload register (TIMx_ARR)(x = 2, 5) 1342
36.5.18 TIMx capture/compare register 1 (TIMx_CCR1)(x = 3, 4) 1342
36.5.19 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2, 5) 1343
36.5.20 TIMx capture/compare register 2 (TIMx_CCR2)(x = 3, 4) 1344
36.5.21 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2, 5) 1345
36.5.22 TIMx capture/compare register 3 (TIMx_CCR3)(x = 3, 4) 1346
36.5.23 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2, 5) 1347
36.5.24 TIMx capture/compare register 4 (TIMx_CCR4)(x = 3, 4) 1348
36.5.25 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2, 5) 1349
36.5.26 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5) 1350
36.5.27 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5) 1351
36.5.28 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5) 1352
36.5.29 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5) 1353

36.5.30 TIMx DMA control register (TIMx_DCR)(x = 2 to 5) 1354
36.5.31 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5) 1355

37 Basic timers (TIM6/TIM7) 1356

37.1 TIM6/TIM7 introduction 1356
37.2 TIM6/TIM7 main features 1356
37.3 TIM6/TIM7 functional description 1357
37.3.1 TIM6/TIM7 block diagram 1357
37.3.2 TIM6/TIM7 internal signals 1357
37.3.3 TIM6/TIM7 clocks 1358
37.3.4 Time-base unit 1358
37.3.5 Counting mode 1360
37.3.6 UIF bit remapping 1366
37.3.7 TIM6/TIM7 DMA requests 1367
37.3.8 Debug mode 1367
37.3.9 TIM6/TIM7 low-power modes 1367
37.3.10 TIM6/TIM7 interrupts 1367
37.4 TIM6/TIM7 registers 1367
37.4.1 TIM6/TIM7 memory map 1368
37.4.2 TIMx control register 1 (TIMx_CR1)(x = 6 to 7) 1368
37.4.3 TIMx control register 2 (TIMx_CR2)(x = 6 to 7) 1370
37.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) 1370
37.4.5 TIMx status register (TIMx_SR)(x = 6 to 7) 1371
37.4.6 TIMx event generation register (TIMx_EGR)(x = 6 to 7) 1371
37.4.7 TIMx counter (TIMx_CNT)(x = 6 to 7) 1371
37.4.8 TIMx prescaler (TIMx_PSC)(x = 6 to 7) 1372
37.4.9 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7) 1372

38 General-purpose timers (TIM15/TIM16) 1373

38.1 TIM15/TIM16 introduction 1373
38.2 TIM15 main features 1373
38.3 TIM16 main features 1374
38.4 TIM15/TIM16 functional description 1375
38.4.1 Block diagram 1375
38.4.2 TIM15/TIM16 pins and internal signals 1376
38.4.3 Time-base unit 1377

38.4.4 Counter modes . . . 1380
38.4.5 Repetition counter . . . 1384
38.4.6 Clock selection . . . 1385
38.4.7 Capture/compare channels . . . 1387
38.4.8 Input capture mode . . . 1389
38.4.9 PWM input mode (only for TIM15) . . . 1391
38.4.10 Forced output mode . . . 1392
38.4.11 Output compare mode . . . 1392
38.4.12 PWM mode . . . 1394
38.4.13 Combined PWM mode (TIM15 only) . . . 1399
38.4.14 Complementary outputs and dead-time insertion . . . 1400
38.4.15 Using the break function . . . 1403
38.4.16 Bidirectional break input . . . 1406
38.4.17 Clearing the tim_ocxref signal on an external event . . . 1408
38.4.18 One-pulse mode . . . 1409
38.4.19 Retriggerable one pulse mode (TIM15 only) . . . 1410
38.4.20 UIF bit remapping . . . 1411
38.4.21 Timer input XOR function (TIM15 only) . . . 1411
38.4.22 External trigger synchronization (TIM15 only) . . . 1412
38.4.23 Slave mode – combined reset + trigger mode (TIM15 only) . . . 1414
38.4.24 Slave mode – combined reset + gated mode (TIM15 only) . . . 1414
38.4.25 Timer synchronization (TIM15) . . . 1415
38.4.26 Using timer output as trigger for other timers (TIM16) . . . 1415
38.4.27 DMA burst mode . . . 1415
38.4.28 TIM15/TIM16 DMA requests . . . 1416
38.4.29 Debug mode . . . 1416
38.5 TIM15/TIM16 low-power modes . . . 1417
38.6 TIM15/TIM16 interrupts . . . 1417
38.7 TIM15/TIM16 memory map . . . 1417
38.8 TIM15 registers . . . 1419
38.8.1 TIM15 control register 1 (TIM15_CR1) . . . 1419
38.8.2 TIM15 control register 2 (TIM15_CR2) . . . 1420
38.8.3 TIM15 slave mode control register (TIM15_SMCR) . . . 1422
38.8.4 TIM15 DMA/interrupt enable register (TIM15_DIER) . . . 1423
38.8.5 TIM15 status register (TIM15_SR) . . . 1424
38.8.6 TIM15 event generation register (TIM15_EGR) . . . 1426

38.8.7 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) 1427
38.8.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) 1428
38.8.9 TIM15 capture/compare enable register (TIM15_CCER) 1431
38.8.10 TIM15 counter (TIM15_CNT) 1434
38.8.11 TIM15 prescaler (TIM15_PSC) 1434
38.8.12 TIM15 auto-reload register (TIM15_ARR) 1435
38.8.13 TIM15 repetition counter register (TIM15_RCR) 1435
38.8.14 TIM15 capture/compare register 1 (TIM15_CCR1) 1436
38.8.15 TIM15 capture/compare register 2 (TIM15_CCR2) 1437
38.8.16 TIM15 break and dead-time register (TIM15_BDTR) 1437
38.8.17 TIM15 timer deadtime register 2 (TIM15_DTR2) 1440
38.8.18 TIM15 input selection register (TIM15_TISEL) 1441
38.8.19 TIM15 alternate function register 1 (TIM15_AF1) 1442
38.8.20 TIM15 alternate function register 2 (TIM15_AF2) 1444
38.8.21 TIM15 DMA control register (TIM15_DCR) 1445
38.8.22 TIM15 DMA address for full transfer (TIM15_DMAR) 1445
38.9 TIM16 registers 1446
38.9.1 TIMx control register 1 (TIMx_CR1)(x = 16) 1446
38.9.2 TIMx control register 2 (TIMx_CR2)(x = 16) 1447
38.9.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16) 1448
38.9.4 TIMx status register (TIMx_SR)(x = 16) 1449
38.9.5 TIMx event generation register (TIMx_EGR)(x = 16) 1450
38.9.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16) 1451
38.9.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16) 1452
38.9.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16) 1454
38.9.9 TIMx counter (TIMx_CNT)(x = 16) 1456
38.9.10 TIMx prescaler (TIMx_PSC)(x = 16) 1456
38.9.11 TIMx auto-reload register (TIMx_ARR)(x = 16) 1457
38.9.12 TIMx repetition counter register (TIMx_RCR)(x = 16) 1457
38.9.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16) 1458
38.9.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16) 1459
38.9.15 TIMx option register 1 (TIMx_OR1)(x = 16) 1462
38.9.16 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16) 1462
38.9.17 TIMx input selection register (TIMx_TISEL)(x = 16) 1463

38.9.18 TIMx alternate function register 1 (TIMx_AF1)(x = 16) 1463
38.9.19 TIMx alternate function register 2 (TIMx_AF2)(x = 16) 1466
38.9.20 TIMx DMA control register (TIMx_DCR)(x = 16) 1466
38.9.21 TIM16 DMA address for full transfer (TIMx_DMAR)(x = 16) 1467

39 Timestamp timer (TIM_TS) 1468
39.1 TIM_TS introduction 1468
39.2 TIM_TS main features 1468
39.3 TIM_TS functional description 1468
39.3.1 TIM_TS block diagram 1468
39.3.2 TIM_TS internal signals 1469
39.3.3 TIM_TS clocks 1469
39.3.4 Time-base unit 1470
39.3.5 Counting mode 1472
39.3.6 UIF bit remapping 1478
39.3.7 TIM_TS DMA requests 1478
39.3.8 Debug mode 1479
39.3.9 TIM_TS low-power modes 1479
39.3.10 TIM_TS interrupts 1479
39.4 TIM_TS registers 1479
39.4.1 TIM_TS memory map 1479
39.4.2 TIM_TS control register 1 (TIM_TS_CR1) 1480
39.4.3 TIM_TS control register 2 (TIM_TS_CR2) 1481
39.4.4 TIM_TS DMA/Interrupt enable register (TIM_TS_DIER) 1481
39.4.5 TIM_TS status register (TIM_TS_SR) 1482
39.4.6 TIM_TS event generation register (TIM_TS_EGR) 1482
39.4.7 TIM_TS counter (TIM_TS_CNT) 1483
39.4.8 TIM_TS prescaler (TIM_TS_PSC) 1483
39.4.9 TIM_TS auto-reload register (TIM_TS_ARR) 1484

40 Real-time clock (RTC) 1485
40.1 Introduction 1485
40.2 RTC main features 1485
40.3 RTC functional description 1486
40.3.1 RTC block diagram 1486
40.3.2 RTC pins and internal signals 1487

40.3.3 Clock and prescalers . . . 1487
40.3.4 Real-time clock and calendar . . . 1488
40.3.5 Programmable alarm . . . 1488
40.3.6 Periodic auto-wakeup . . . 1489
40.3.7 RTC initialization and configuration . . . 1489
40.3.8 Reading the calendar . . . 1491
40.3.9 Resetting the RTC . . . 1492
40.3.10 RTC synchronization . . . 1492
40.3.11 RTC smooth digital calibration . . . 1492
40.3.12 Debug mode . . . 1494
40.4 RTC low-power modes . . . 1495
40.5 RTC interrupts . . . 1495
40.6 RTC registers . . . 1495
40.6.1 RTC memory map . . . 1495
40.6.2 RTC time register (RTC_TR) . . . 1496
40.6.3 RTC date register (RTC_DR) . . . 1497
40.6.4 RTC sub second register (RTC_SSR) . . . 1497
40.6.5 RTC initialization control and status register (RTC_ICSR) . . . 1498
40.6.6 RTC prescaler register (RTC_PRER) . . . 1499
40.6.7 RTC wakeup timer register (RTC_WUTR) . . . 1500
40.6.8 RTC control register (RTC_CR) . . . 1500
40.6.9 RTC write protection register (RTC_WPR) . . . 1502
40.6.10 RTC calibration register (RTC_CALR) . . . 1503
40.6.11 RTC shift control register (RTC_SHIFTR) . . . 1504
40.6.12 RTC alarm A register (RTC_ALRMAR) . . . 1505
40.6.13 RTC alarm A sub second register (RTC_ALRMASSR) . . . 1506
40.6.14 RTC status register (RTC_SR) . . . 1506
40.6.15 RTC masked interrupt status register (RTC_MISR) . . . 1507
40.6.16 RTC status clear register (RTC_SCR) . . . 1507

41 Universal asynchronous receiver transmitter (UART) . . . 1509
41.1 UART introduction . . . 1509
41.2 UART main features . . . 1509
41.3 UART extended features . . . 1510
41.4 UART implementation . . . 1510
41.5 UART functional description . . . 1511

41.5.1 UART block diagram . . . 1511
41.5.2 UART signals . . . 1512
41.5.3 UART character description . . . 1512
41.5.4 UART FIFOs and thresholds . . . 1514
41.5.5 UART transmitter . . . 1514
41.5.6 UART receiver . . . 1518
41.5.7 UART baud rate generation . . . 1525
41.5.8 Tolerance of the UART receiver to clock deviation . . . 1526
41.5.9 UART Auto baud rate detection . . . 1528
41.5.10 UART multiprocessor communication . . . 1530
41.5.11 UART Modbus communication . . . 1532
41.5.12 UART parity control . . . 1533
41.5.13 UART LIN (local interconnection network) mode . . . 1534
41.5.14 UART single-wire Half-duplex communication . . . 1536
41.5.15 UART receiver timeout . . . 1536
41.5.16 UART IrDA SIR ENDEC block . . . 1537
41.5.17 Continuous communication using UART and DMA . . . 1539
41.5.18 UART low-power management . . . 1542
41.6 UART interrupts . . . 1545
41.7 UART registers . . . 1547
41.7.1 UART memory map . . . 1547
41.7.2 UART control register 1 [alternate] (UART_CR1) . . . 1548
41.7.3 UART control register 1 [alternate] (UART_CR1) . . . 1551
41.7.4 UART control register 2 (UART_CR2) . . . 1555
41.7.5 UART control register 3 (UART_CR3) . . . 1558
41.7.6 UART baud rate register (UART_BRR) . . . 1563
41.7.7 UART guard time and prescaler register (UART_GTPR) . . . 1563
41.7.8 UART receiver timeout register (UART_RTOR) . . . 1564
41.7.9 UART request register (UART_RQR) . . . 1565
41.7.10 UART interrupt and status register [alternate] (UART_ISR) . . . 1566
41.7.11 UART interrupt and status register [alternate] (UART_ISR) . . . 1571
41.7.12 UART interrupt flag clear register (UART_ICR) . . . 1576
41.7.13 UART receive data register (UART_RDR) . . . 1578
41.7.14 UART transmit data register (UART_TDR) . . . 1578
41.7.15 UART prescaler register (UART_PRESC) . . . 1579

42 Serial peripheral interface / integrated interchip sound (SPI/I2S) . 1580

42.1 Introduction . . . 1580
42.2 SPI main features . . . 1580
42.3 I2S main features . . . 1581
42.4 SPI/I2S implementation . . . 1581
42.5 SPI functional description . . . 1582
42.5.1 General description . . . 1582
42.5.2 Communications between one master and one slave . . . 1583
42.5.3 Standard multi-slave communication . . . 1585
42.5.4 Multi-master communication . . . 1586
42.5.5 Slave select (NSS) pin management . . . 1587
42.5.6 Communication formats . . . 1588
42.5.7 Configuration of SPI . . . 1590
42.5.8 Procedure for enabling SPI . . . 1591
42.5.9 Data transmission and reception procedures . . . 1591
42.5.10 SPI status flags . . . 1601
42.5.11 SPI error flags . . . 1602
42.5.12 NSS pulse mode . . . 1603
42.5.13 TI mode . . . 1603
42.5.14 CRC calculation . . . 1604
42.6 SPI interrupts . . . 1606
42.7 I2S functional description . . . 1607
42.7.1 I2S general description . . . 1607
42.7.2 Supported audio protocols . . . 1608
42.7.3 Startup description . . . 1615
42.7.4 Clock generator . . . 1617
42.7.5 I^2S master mode . . . 1620
42.7.6 I^2S slave mode . . . 1621
42.7.7 I2S status flags . . . 1623
42.7.8 I2S error flags . . . 1624
42.7.9 DMA features . . . 1625
42.8 I2S interrupts . . . 1625
42.9 SPI and I2S registers . . . 1626
42.9.1 SPI and I2S memory map . . . 1626
42.9.2 SPI control register 1 (SPIx_CR1) . . . 1626
42.9.3 SPI control register 2 (SPIx_CR2) . . . 1628
42.9.4 SPI status register (SPIx_SR) . . . 1630

42.9.5 SPI data register (SPIx_DR) . . . 1632
42.9.6 SPI CRC polynomial register (SPIx_CRCPR) . . . 1632
42.9.7 SPI Rx CRC register (SPIx_RXCRCR) . . . 1633
42.9.8 SPI Tx CRC register (SPIx_TXCRCR) . . . 1633
42.9.9 SPIx_I2S configuration register (SPIx_I2SCFGR) . . . 1634
42.9.10 SPIx_I2S prescaler register (SPIx_I2SPR) . . . 1635

43 Inter-integrated circuit (I2C) interface . . . 1637
43.1 Introduction . . . 1637
43.2 I2C main features . . . 1637
43.3 I2C implementation . . . 1638
43.4 I2C functional description . . . 1638
43.4.1 I2C block diagram . . . 1639
43.4.2 I2C pins and internal signals . . . 1640
43.4.3 I2C clock requirements . . . 1640
43.4.4 Mode selection . . . 1640
43.4.5 I2C initialization . . . 1641
43.4.6 Software reset . . . 1646
43.4.7 Data transfer . . . 1647
43.4.8 I2C slave mode . . . 1649
43.4.9 I2C master mode . . . 1658
43.4.10 I2C_TIMINGR register configuration examples . . . 1669
43.4.11 SMBus specific features . . . 1671
43.4.12 SMBus initialization . . . 1674
43.4.13 SMBus: I2C_TIMEOUTR register configuration examples . . . 1676
43.4.14 SMBus slave mode . . . 1677
43.4.15 Error conditions . . . 1683
43.4.16 DMA requests . . . 1685
43.4.17 Debug mode . . . 1686
43.5 I2C low-power modes . . . 1686
43.6 I2C interrupts . . . 1687
43.7 I2C registers . . . 1687
43.7.1 I2C memory map . . . 1687
43.7.2 I2C control register 1 (I2C_CR1) . . . 1688
43.7.3 I2C control register 2 (I2C_CR2) . . . 1691
43.7.4 I2C own address 1 register (I2C_OAR1) . . . 1693

43.7.5 I2C own address 2 register (I2C_OAR2) 1694
43.7.6 I2C timing register (I2C_TIMINGR) 1695
43.7.7 I2C timeout register (I2C_TIMEOUTR) 1696
43.7.8 I2C interrupt and status register (I2C_ISR) 1697
43.7.9 I2C interrupt clear register (I2C_ICR) 1699
43.7.10 I2C PEC register (I2C_PECR) 1700
43.7.11 I2C receive data register (I2C_RXDR) 1701
43.7.12 I2C transmit data register (I2C_TXDR) 1701

44 Independent watchdog (IWDG) 1702
44.1 Introduction 1702
44.2 IWDG main features 1702
44.3 IWDG functional description 1702
44.3.1 IWDG block diagram 1702
44.3.2 Window option 1703
44.3.3 Hardware watchdog 1704
44.3.4 Register access protection 1704
44.3.5 Debug mode 1704
44.4 IWDG registers 1705
44.4.1 IWDG memory map 1705
44.4.2 IWDG key register (IWDG_KR) 1705
44.4.3 IWDG prescaler register (IWDG_PR) 1706
44.4.4 IWDG reload register (IWDG_RLR) 1707
44.4.5 IWDG status register (IWDG_SR) 1708
44.4.6 IWDG window register (IWDG_WINR) 1709

45 System window watchdog (WWDG) 1710
45.1 Introduction 1710
45.2 WWDG main features 1710
45.3 WWDG functional description 1710
45.3.1 WWDG block diagram 1711
45.3.2 Enabling the watchdog 1711
45.3.3 Controlling the down-counter 1711
45.3.4 How to program the watchdog timeout 1711
45.3.5 Debug mode 1712
45.4 WWDG interrupts 1713

45.5 WWDG registers 1713
45.5.1 WWDG memory map 1713
45.5.2 WWDG control register (WWDG_CR) 1713
45.5.3 WWDG configuration register (WWDG_CFR) 1714
45.5.4 WWDG status register (WWDG_SR) 1715

46 CAN subsystem 1716
46.1 Introduction 1716
46.2 Features 1718
46.3 Controller area network (M_CAN) 1718
46.4 DMA interface unit (DMU) 1718
46.5 Timestamping unit (TSU) 1718
46.6 CAN RAM arbiter 1718
46.6.1 Features 1719
46.6.2 Functional overview using examples 1719
46.7 SRAM interface and memory organization 1719
46.7.1 ECC controller 1719
46.8 CAN nodes 1720
46.9 External Timestamp 1721
46.10 External signal description 1722
46.11 Shared memory map 1722

47 Controller area network (M_CAN) 1724
47.1 Overview 1724
47.1.1 Features 1725
47.1.2 Block diagram 1726
47.1.3 Dual clock sources 1727
47.1.4 Dual interrupt lines 1727
47.2 Memory map and register descriptions 1727
47.2.1 Hardware reset description 1727
47.2.2 Register map 1727
47.2.3 Registers 1730
47.2.4 Message RAM 1776
47.2.5 M_CAN functional description 1787
47.2.6 Timestamp generation 1798
47.2.7 Timeout counter 1798

47.2.8 Rx handling . . . 1799
47.2.9 Dedicated Rx buffers . . . 1806
47.2.10 Debug on CAN support . . . 1807
47.2.11 Tx handling . . . 1809
47.2.12 FIFO acknowledge handling . . . 1814

48 DMA interface unit (DMU) . . . 1815
48.1 Overview . . . 1815
48.1.1 M_CAN message handling (without DMU) . . . 1815
48.1.2 Message handling with DMU . . . 1815
48.1.3 Block diagram . . . 1816
48.2 Memory map and register definitions . . . 1817
48.2.1 Hardware reset description . . . 1817
48.2.2 Memory map . . . 1817
48.2.3 Registers . . . 1818
48.3 Functional description . . . 1832
48.3.1 DMU TX element (0x200–0x244) . . . 1832
48.3.2 DMU RX element (0x280–0x2C4 and 0x300–0x344) . . . 1832
48.3.3 TX event element (0x380–0x388) . . . 1833
48.3.4 Exception recovery at DMU virtual buffer . . . 1834
48.3.5 Transferring timestamps from TSU . . . 1834
48.3.6 M_CAN in CCE mode . . . 1835
48.3.7 DMU service request signals . . . 1835
48.3.8 Software debugging support . . . 1836

49 Timestamping unit (TSU) . . . 1838
49.1 Introduction . . . 1838
49.2 Memory map and register descriptions . . . 1838
49.2.1 Memory map . . . 1838
49.2.2 Registers . . . 1838
49.3 Functional description . . . 1844
49.3.1 M_CAN configuration . . . 1844
49.3.2 TSU operation . . . 1844
49.4 TSU integration . . . 1846
49.4.1 TSU with internal 32-bit counter as timebase . . . 1847
49.4.2 TSU without internal timebase . . . 1847

49.5 New features of M_CAN 1847
49.5.1 Generic parameter 1847
49.5.2 Additional interface signals 1848
49.5.3 Timestamp control 1848
49.5.4 Modification of Tx buffer element 1849
49.5.5 Modification of Rx Buffer and FIFO element 1849
49.5.6 Modification of Tx event FIFO element 1849
49.5.7 Modification of standard message ID Filter Element 1849
49.5.8 Modification of extended message ID Filter Element 1850

50 Hardware semaphore (HSEM2) 1851
50.1 Hardware semaphore introduction 1851
50.2 Hardware semaphore main features 1851
50.3 HSEM2 block diagram 1852
50.4 HSEM2 functional description 1852
50.4.1 HSEM2 lock procedures 1852
50.4.2 HSEM2 write/read/readlock register address 1854
50.4.3 HSEM2 clear procedures 1854
50.4.4 HSEM2 COREID semaphore clear 1854
50.4.5 HSEM2 interrupts 1855
50.4.6 AHB bus master ID verification 1856
50.5 HSEM2 registers 1857
50.5.1 HSEM2 memory map 1857
50.5.2 HSEM2 register semaphore x (HSEM2_Rx) 1858
50.5.3 HSEM2 read lock register semaphore x (HSEM2_RLRx) 1859
50.5.4 HSEM2 interrupt enable register (HSEM2_IERn) (n=0 to 2) 1860
50.5.5 HSEM2 interrupt clear register (HSEM2_ICRn) (n=0 to 2) 1860
50.5.6 HSEM2 interrupt status register (HSEM2_ISRn) (n=0 to 2) 1860
50.5.7 HSEM2 interrupt status register (HSEM2_MISRn) (n=0 to 2) 1861
50.5.8 HSEM2 clear register (HSEM2_CR) 1861
50.5.9 HSEM2 interrupt clear register (HSEM2_KEYR) 1862

51 Fault collection and control unit (FCCU) 1863
51.1 Introduction 1863
51.2 Features 1863
51.2.1 Standard feature 1864

51.3 Block diagram . . . 1864
51.4 Signal description . . . 1865
51.4.1 Pinout . . . 1865
51.4.2 IPS bus interface . . . 1866
51.5 Functional description . . . 1866
51.5.1 Definitions . . . 1866
51.5.2 FSM description . . . 1867
51.5.3 Reset interface . . . 1868
51.5.4 Fault priority scheme and nesting . . . 1869
51.5.5 Fault recovery . . . 1870
51.5.6 NMI/WKPU interface . . . 1872
51.5.7 FAULT interface . . . 1873
51.5.8 STCU interface . . . 1874
51.5.9 EOUT interface . . . 1875
51.5.10 Error signal flow for RF case . . . 1880
51.6 Register description . . . 1882
51.6.1 FCCU control register (FCCU_CTRL) . . . 1884
51.6.2 FCCU CTRL key register (FCCU_CTRLK) . . . 1886
51.6.3 FCCU configuration register (FCCU_CFG) . . . 1887
51.6.4 FCCU RF configuration register n (FCCU_RF_CFGn) . . . 1889
51.6.5 FCCU RFS configuration register n (FCCU_RFS_CFGn) . . . 1890
51.6.6 FCCU RF status register n (FCCU_RF_Sn) . . . 1891
51.6.7 FCCU RF key register (FCCU_RFK) . . . 1893
51.6.8 FCCU RF enable register n (FCCU_RF_En) . . . 1894
51.6.9 FCCU RF timeout enable register n (FCCU_RF_TOEn) . . . 1895
51.6.10 FCCU RF timeout register (FCCU_RF_TO) . . . 1896
51.6.11 FCCU CFG timeout register (FCCU_CFG_TO) . . . 1897
51.6.12 FCCU IO control register (FCCU_EINOUT) . . . 1898
51.6.13 FCCU status register (FCCU_STAT) . . . 1899
51.6.14 FCCU NA freeze status register (N2AF_STATUS) . . . 1901
51.6.15 FCCU AF freeze status register (A2FF_STATUS) . . . 1901
51.6.16 FCCU NF freeze status register (N2FF_STATUS) . . . 1902
51.6.17 FCCU FA freeze status register (F2A_STATUS) . . . 1903
51.6.18 FCCU RF fake register (FCCU_RFF) . . . 1904
51.6.19 FCCU IRQ status register (FCCU_IRQ_STAT) . . . 1905
51.6.20 FCCU IRQ enable register (FCCU_IRQ_EN) . . . 1906
51.6.21 FCCU XTMR register (FCCU_XTMR) . . . 1907

51.6.22 FCCU transient register (FCCU_TRANS_LOCK) 1908
51.6.23 FCCU permanent lock register (FCCU_PERMNT_LOCK) 1909
51.6.24 FCCU delta T register (FCCU_DELTA_T) 1909
51.6.25 FCCU IRQ alarm enable register n (FCCU_IRQ_ALARM_ENn) . . . 1910
51.6.26 FCCU NMI enable registers n (FCCU_NMI_ENn) 1911
51.6.27 FCCU EOUT signaling enable registers n (FCCU_EOUT_SIG_ENn) 1912
51.6.28 FCCU NMI2 enable registers n (FCCU_NMI2_ENn) 1913
51.7 FCCU output supervision unit (FOSU) 1914
51.8 Use cases and limitations 1916
51.8.1 Misconfigurations 1916
51.8.2 Recommendations to configure FCCU 1916

52 Collective error manager (CEM) 1917
52.1 Introduction 1917
52.1.1 Overview 1917
52.1.2 Block diagram 1917
52.1.3 Features 1918
52.2 System interfaces 1918
52.2.1 System block diagram 1919
52.2.2 Bus interface 1919
52.2.3 IO interface 1919
52.2.4 Interrupt interface 1919
52.2.5 DMA interface 1920
52.2.6 Clock interface 1920
52.2.7 Reset interface 1920
52.2.8 Safety fault interface 1920
52.3 Registers 1920
52.3.1 Memory map 1920
52.3.2 Register description 1921
52.4 Functional description 1926
52.4.1 CEM functionality 1926
52.4.2 CEM internal registers 1926
52.4.3 CEM Programmable register 1927
52.4.4 Clock and reset 1927
52.4.5 Asynchronous fault capture 1927
52.4.6 Debug mode behavior 1928

52.5 Safety mechanism 1928
52.6 Security 1928
52.7 Programming guidelines 1928

53 Memory error management unit 2 (MEMU2) 1932
53.1 Introduction 1932
53.2 Features 1933
53.3 Block diagram 1933
53.4 Design overview 1934
53.4.1 Memory to MEMU agent 1934
53.4.2 Error processing block 1935
53.4.3 Reporting table 1936
53.4.4 Register block 1936
53.4.5 ECC filtered reaction 1937
53.5 External signal description 1940
53.6 Memory map and registers definition 1940
53.6.1 Memory map 1940
53.6.2 Register descriptions 1944
53.7 Functional description 1975
53.7.1 Initializing MEMU2 1975
53.7.2 Reading the reporting table 1976
53.7.3 Handling overflows 1977

54 Indirect memory access (IMA) 1978
54.1 Introduction 1978
54.2 Accessing memory through the IMA module 1978
54.2.1 Features 1979
54.2.2 Block diagram 1979
54.2.3 IMA module memory map and registers 1980
54.2.4 IMA module functional description 1987
54.2.5 Application information 1988

55 Self-test control unit (STCU3) 1989
55.1 Introduction 1989
55.1.1 Overview 1989
55.1.2 Block diagram 1989

55.2 Features . . . 1991
55.3 Functional description . . . 1992
55.3.1 FSM description . . . 1992
55.3.2 DCF interface . . . 1993
55.3.3 Reset management . . . 1993
55.3.4 BIST scheduling . . . 1993
55.3.5 PLL interface . . . 1995
55.3.6 Interrupt interface . . . 1995
55.3.7 Fault collector interface . . . 1995
55.3.8 Watchdog support . . . 1995
55.3.9 CRC support . . . 1996
55.4 Programming guidelines . . . 1996
55.4.1 Offline self-test sequence . . . 1996
55.4.2 Bypass user mode . . . 1998
55.5 Memory map and register description . . . 1998
55.5.1 Memory map . . . 1998
55.5.2 Register description . . . 1999

56 Tamper detection module (TDM) . . . 2025
56.1 Overview . . . 2025
56.2 Features . . . 2026
56.3 External signal description . . . 2026
56.4 DCF client . . . 2026
56.4.1 Diary base address (DBA) DCF client . . . 2027
56.4.2 Tamper region override (TO) DCF client . . . 2028
56.4.3 One time programmable enable (OTPEN0) DCF client . . . 2028
56.4.4 One time programmable enable (OTPEN1) DCF client . . . 2029
56.4.5 One time programmable enable (OTPEN2) DCF client . . . 2030
56.4.6 One time programmable enable (OTPEN3) DCF client . . . 2031
56.4.7 Tamper region assignment DCF client (TDRn_LOCK0) . . . 2031
56.4.8 Tamper region assignment DCF client (TDRn _LOCK1) . . . 2032
56.4.9 Tamper region assignment DCF client (TDRn_LOCK2) . . . 2033
56.4.10 Tamper region assignment DCF client (TDRn_LOCK3) . . . 2034
56.5 Memory map and registers . . . 2034
56.5.1 TDR status register (TDM_TDRSR) . . . 2035
56.5.2 Last flash programmed address register (TDM_LFPAR) . . . 2036

56.5.3 Diary base address (TDM_DBA) 2037
56.5.4 Software tamper override key region (TDM_STO_KEYn) 2038
56.6 Functional description 2038
56.6.1 Flash erase counter and tamper detection 2038
56.6.2 Assigning blocks to tamper detect regions (TDR) 2040
56.6.3 Diary 2040
56.6.4 Specifying the diary base address 2041
56.6.5 Diary base address verification 2042
56.6.6 One time programmable (OTP) 2043

Appendix A Acronyms and abbreviations 2044

Revision history 2048

1 Preface

1.1 Document organization

This document includes chapters that are divided into parts:

- Part I includes chapters that describe the device as a whole or provide device-specific information.
- Part II includes chapters that describe the functionality of the individual modules on the device.

1.2 Register conventions

Figure 1 and *Table 1* provide the register conventions used throughout this manual.

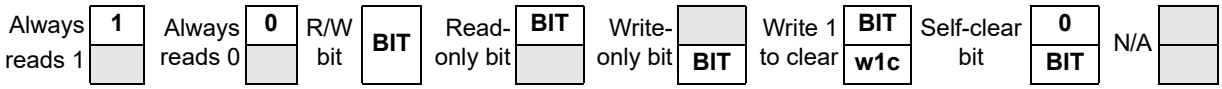


Figure 1. Key to register fields

Table 1. Register conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register field types	
R, r	Read only. Writing this bit has no effect.
W, wo	Write only
R/W, rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c, rc_w1, rc1	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset values	
0	Resets to zero
1	Resets to one
—	Undefined at reset

1.3 Acronyms and abbreviations

Appendix A: Acronyms and abbreviations lists acronyms and abbreviations used in this document.

1.4 Reference documents

In addition to this reference manual, the following documents provide additional information on the operation of the SR5E1x:

- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture

Note: For information on the CoreSight™ and Arm® Cortex®-M7 cores, refer to the CoreSight™ Architecture Specification v2.0 and Arm® Cortex®-M7 technical reference manual, available from the www.arm.com website.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

2 Introduction

The SR5E1x belongs to a new generation of microcontroller family built on Arm® Cortex®-M7 cores, bringing new level of performance, safety, and security for real-time automotive applications.

The device especially targets analog performance platforms and fast digital control applications, such as in the fields of:

- DC/DC converter
- On-board charger
- Advanced motor control

Note: The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

2.1 Overall architecture

The architecture is split between one cluster domain that performs various computational and control functions and a HSM domain as shown in *Figure 2: Block diagram*. All components are connected through a crossbar.

Cluster domain has 2 Arm® Cortex®-M7 cores that can work either in decoupled or lockstep modes. Each core is identical in implementation and contains the same amount of caches/Tightly Coupled Memory (TCM) memories that can be fully accessed even when cores are working in lock-step modes.

The HSM includes one Arm® Cortex®-M0+.

Table 2. Processor cores

Core	Type
Cortex®-M7 Cluster 0	
Main Core_1 (Boot)	Arm® Cortex®-M7
Main/Checker Core_2	Arm® Cortex®-M7
Hardware Security Module	
Arm® Cortex®-M0+	

2.1.1 Cortex®-M7

The Arm® Cortex®-M7 with double-precision FPU processor is a highly efficient high-performance core featuring:

- Six-stage dual-issue pipeline
- Dynamic branch prediction
- Harvard architecture with L1 caches (8 Kbytes of I-cache and 16 Kbytes of D-cache)
- 64-bit AXI4 interface
- 64-bit ITCM interface
- 2x32-bit DTCM interfaces

The following memory interfaces are supported:

- Separate Instruction and Data buses (Harvard Architecture) to optimize CPU latency
- Tightly Coupled Memory (TCM) interface designed for fast and deterministic SRAM accesses
- AXI Bus interface to optimize Burst transfers
- Dedicated low-latency AHB-Lite peripheral bus (AHBP) to connect to peripherals.

The processor supports a set of DSP instructions, which allow efficient signal processing and complex algorithm execution.

It also supports single and double precision FPU (floating-point unit) that speed up software development by using metalanguage development tools, while avoiding saturation.

2.1.2 Cortex®-M0+

The Arm® Cortex®-M0+ processor is a low-power processor intended for deeply embedded applications that require fast interrupt response features. It features low gate count, low interrupt latency, and low-cost debug.

2.1.3 Memory hierarchy

The SR5E1x comprises different levels of memory hierarchy:

- Memory private to each core: instruction cache, data cache and TCMs. TCMs are unified. They provide fast access and have the most deterministic memory access timing.
- System RAM.

2.2 Features

- AEC-Q100 automotive qualification
- SR5 high-performance analog MCUs offering:
  - Digital and analog high-frequency control requested by new wide-bandgap technologies (silicon carbide and GaN)
  - Superior real-time and functional safety performance (ASIL-D capability)
  - Built-in fast and cost optimized over-the-air (OTA) reprogramming capability (with built-in dual image storage)
  - High-speed security cryptographic services (HSM)

Cores

- 2x 32-bit Arm® Cortex®-M7 with double-precision FPU, L1 cache and DSP instructions running at up to 306.7 MHz to reach 1284 DMIPS/2.14 DMIPS/MHz/Core (Dhrystone 2.1)
  - Split-lock configuration, allowing either two cores in parallel or one core in lockstep configuration
- 2 DMA engines in lockstep configuration

Memories

- Up to 2 MB on-chip flash memory with read while write support
  - 1920 KB code flash memory split in two banks allowing 960 KB OTA (over-the-air) reprogramming support
  - 160 KB HSM dedicated code Flash memory
- 96 KB data Flash memory (64 KB + 32 KB dedicated to HSM)
- 488 KB on-chip general-purpose SRAM:
  - 2× 32 KB instruction TCM + 2x 64 KB data TCM
  - 256 KB system RAM
  - 40 KB HSM dedicated system RAM

Security: hardware security module (HSM)

- On-chip high performance security module with EVITA medium support with dedicated RAM and flash
- Based on Cortex®-M0+ cores running up to 153.35 MHz (half of the system frequency)
- Hardware accelerator for symmetric cryptography

Safety

- Comprehensive new generation ASIL-D safety concept
- State of the art safety measures at all level of the architecture for most efficient implementation of ISO26262 ASIL-D functionalities
- FCCU for collection and reaction to failure notifications with enhanced configurability
- Memory error management unit (MEMU) for collection and reporting of error events in memories
- Cyclic redundancy check (CRC) unit

Enhanced peripherals for fast control loop capability

- 12 timers:
  - 2× HRTIM (Hi-Resolution and complex waveform builder) in total: 12× 16-bit counters, 104 ps resolution, 24 PWM
  - 2× 16-bit 6-channel advanced control timers in total, with up to 12× PWM
  - 2× 32-bit general purpose timers in total, with up to 8× IC/OC/PWM or pulse counter and quadrature encoder input
  - 4× 16-bit general purpose timers in total, with up to 11× PWM, 2 of which paired
  - 2× 16-bit basic timers
- Enhanced analog-to-digital converter system with:
  - 5 separate 12-bit SAR analog converters, 8 channels each. Sampling rate up to 2.5 MSPS in single mode, 5 MSPS in dual mode
  - 2 separate 16-bit sigma-delta analog converters
- 12-bit digital-to-analog converters (DAC)
  - 2 buffered external channels 1 MSPS
  - 8 unbuffered internal channels 15 MSPS
- 8 rail-to-rail analog comparators, 50 ns propagation delay
- Hardware accelerator
  - 1× CORDIC for trigonometric functions acceleration

Communication interfaces

- 4 modular controller area network (MCAN) modules, all supporting flexible data rate (ISO CAN-FD)
- 3 UART modules with LIN functionality
- 4 serial peripheral interface (SPI) modules, 2 multiplexed with I²S interfaces
- 2 I²C modules

Advanced debug and trace for high performance automotive application development

- Built around Arm® CoreSight™-600
- Debug interface: Arm® CoreSight™ JTAG (IEEE 1149.1) or SWD
- 4 KB Embedded Trace FIFO for both on- and off-chip tracing
- Trace port for off-chip tracing: parallel trace port configurable from one to eight data lines

Others

- Power-efficiency management, through separate power modes for any selected cores, peripherals, or memories
- Boot assist flash (BAF) supports factory programming using a serial loader through CAN or UART
- Junction temperature range -40 °C to 150 °C
- Integrated power supply scheme:
  - Integrated internal SMPS regulator
  - 3.3 V supply & GPIOs

2.3 Features list

The following table lists a summary of major features for the SR5E1x. A detailed description of the functionality provided by each on-chip module is given later in this document.

Table 3. Features list

Feature	Description
Cortex®-M7 cluster 0	
Number of cores / checker cores	2 decoupled cores in split mode / 1 core with checker core in lock mode
Nominal frequency	300 MHz
Tightly coupled memory (TCM)	2 × 32 KB instruction in split mode or 1x 64 KB instruction in lock mode 2 × 64 KB data in split mode or 1x 128 KB data in lock mode
Floating point unit	Single and double precision
Cache	8 KB instruction (2 ways) 16 KB data (4 ways)
System memories	
On-chip code flash memory	2 MB
On-chip data flash memory	64 KB
Built-in memory replication for over the air (OTA) reprogramming	Up to 960 KB Flash available
System RAM	2 × 128 KB (not including HSM dedicated data flash - see HSM)
Others	
Multichannel DMA (paired in lockstep)	2 DMA engines, 8 streams each
Interrupt broadcasting system in lockstep	Up to 190 sources
Watchdogs	2 independent and 2 window watchdogs
Security: Hardware security module (HSM)	
Core	Cortex®-M0+ @ 150 MHz, as half device frequency
C3 cryptography engine	Symmetric: – AES-128/256, ECB, CBC, CMAC, GCM – TRNG
Dedicated Flash memory	160 KB
Dedicated system RAM	40 KB
Dedicated data Flash	32 KB
Peripheral, IOs	
Timer modules	
High resolution timer	2 modules, 6 × 16-bit channels each, 104 ps resolution Up to 24× PWM signals (or 12× paired)

Table 3. Features list (continued)

Feature	Description
Advance control timer	2 modules, 16-bit timer Up to 8 input capture, 12 output compare (8 of which paired)
General purpose timer	2 modules, 32-bit timer. Up to 8 input capture / output compare 4 modules, 16-bit timer. Up to 11 input capture / output compare (2 of which paired)
Basic timer	2 modules, 16-bit timer
Enhanced analog-to-digital converter system	
12-bit SAR analog converters	5 modules, 8 channels each Fast conversion, up to 2.5 MSPS in single mode, 5 MSPS in dual mode
16-bit sigma-delta analog converters	2 modules, 2 channels each (available only in QFP176 and QFP144 packages) Output conversion rate of 333 ksps (OSR = 24)
12-bit analog comparators	8 modules, rail-to-rail, 50 ns propagation delay
12-bit digital-to-analog converters	2 buffered external channels, 1 MSPS 8 unbuffered internal digital-to-analog channels, 15 MSPS
HW accelerator	
CORDIC (for trigonometric functions acceleration)	1 module
Communication interfaces	
UART-modules (with LIN function)	3
MCAN supporting CAN-FD according to ISO 11898-1 2015	4 CAN shared message RAM: 4 KB / MCAN (16 KB in total)
Serial peripheral interface (SPI)	4
I2C	2
SW development/emulation features	
Arm® CoreSight™-600 libs	CoreSight™-600 libs for trace links, trace sink and control components CoreSight™-400 libs for debug and trace source components
Debug interfaces	Arm® CoreSight™-600 compliant – Debug port (JTAG+SWD)
Trace types	Cortex®-M7 instruction and data trace
Off-chip trace	Arm® CoreSight™ parallel trace port 1 to 8 data lines
Advance cross trigger and performance measurement	CoreSight-600™ CTI & CTM
Timestamp distribution	Arm® CoreSight™ timestamp generator

Table 3. Features list (continued)

Feature	Description
Security	Arm® CoreSight™ authentication Password challenge with HSM
Debug controller	External tool-host CPU mailbox Host-based debugging Debug-under-reset
Others	
Low power modes	Clock gating management for selected cores, peripherals, and/or memories Smart wake-up mechanisms through events or interrupts
Temperature sensor	Yes
Self test controller	Yes
PLL	2 individual PLLs: 1 with stable clock source for peripherals and 1 supporting frequency modulation for cores
Power supply	Single internal SMPS regulator for 3.3 V supply and GPIOs
Boot assisted Flash (BAF)	Supports factory programming using a serial loader through the asynchronous CAN or UART
CRC channel(s)	1

2.4 Block diagram

The figure below shows the top-level block diagram.

Figure 2. Block diagram

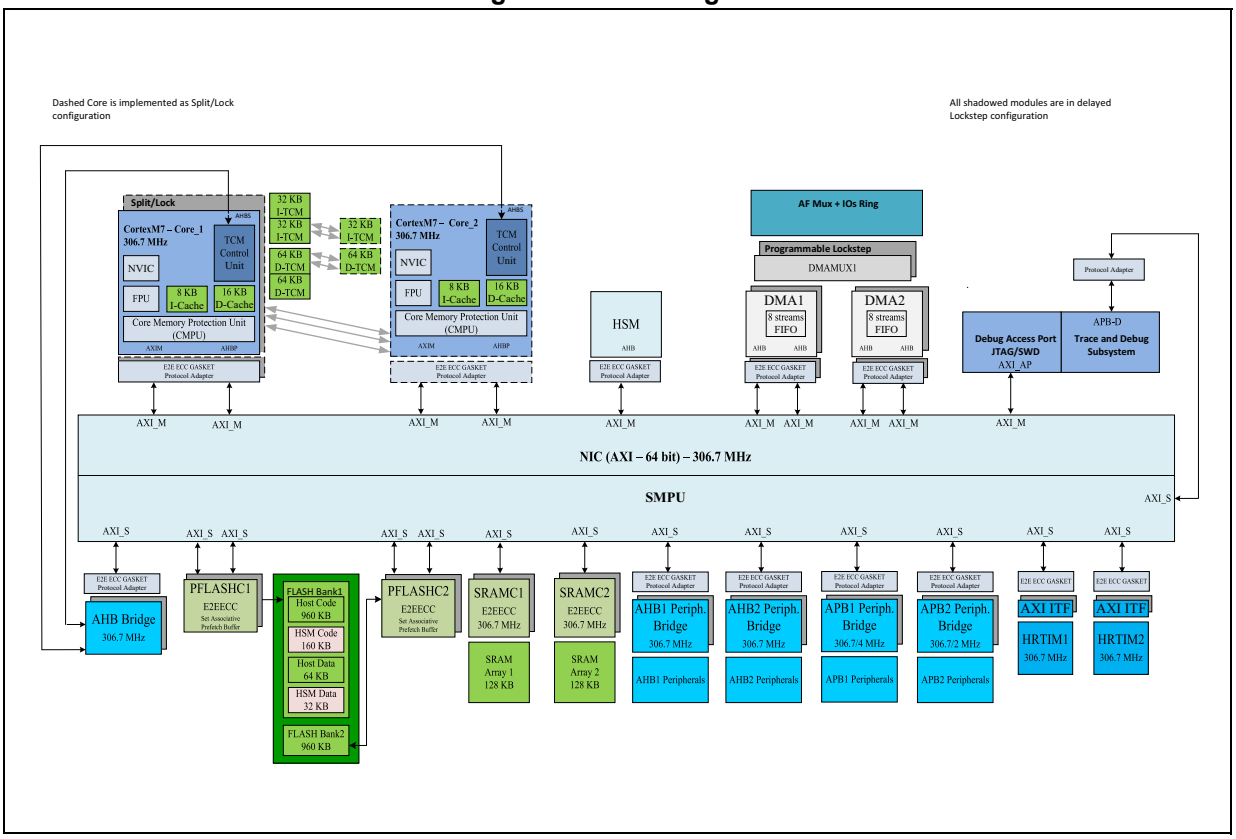
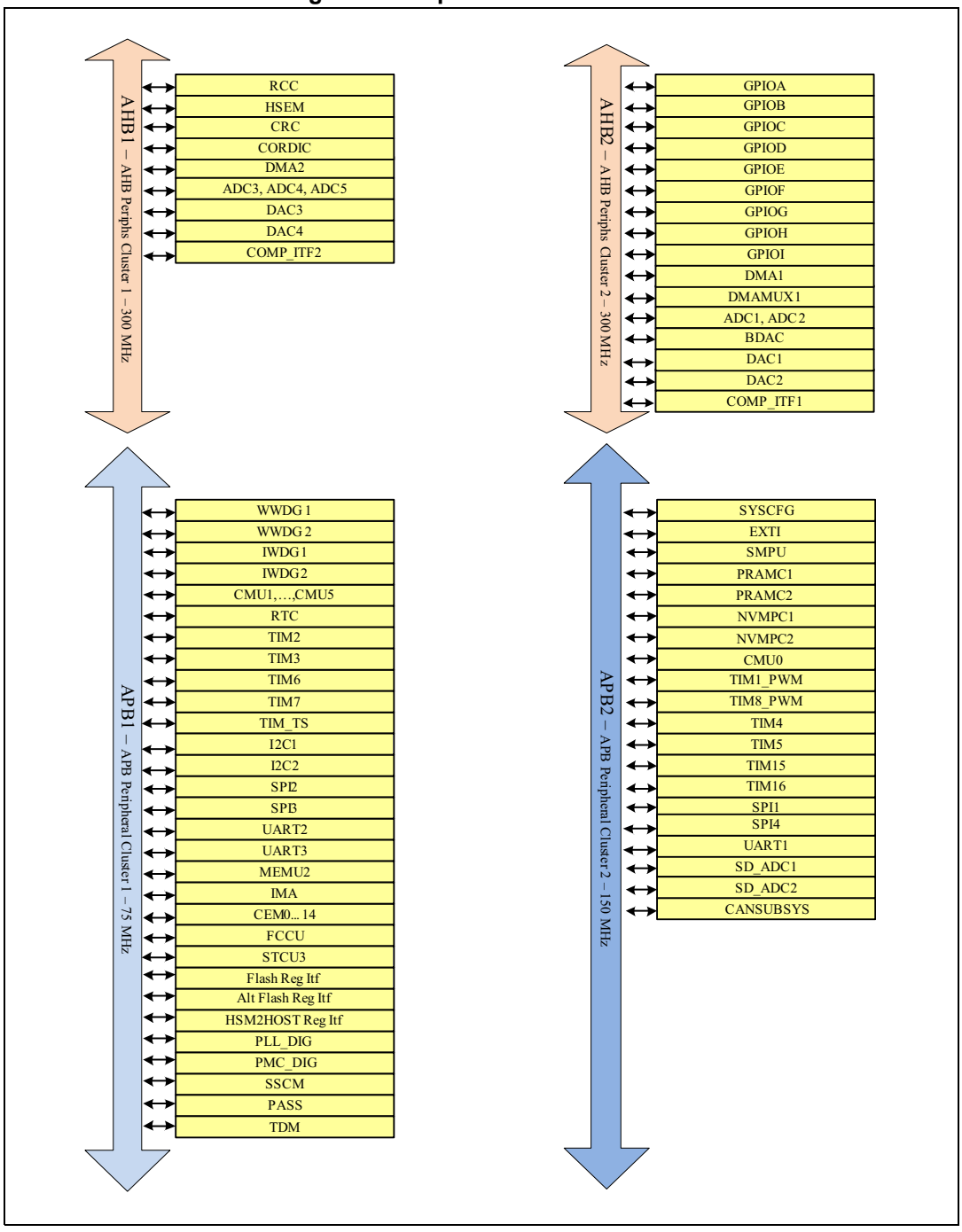


Figure 3. Peripheral cluster source



3 System and memory overview

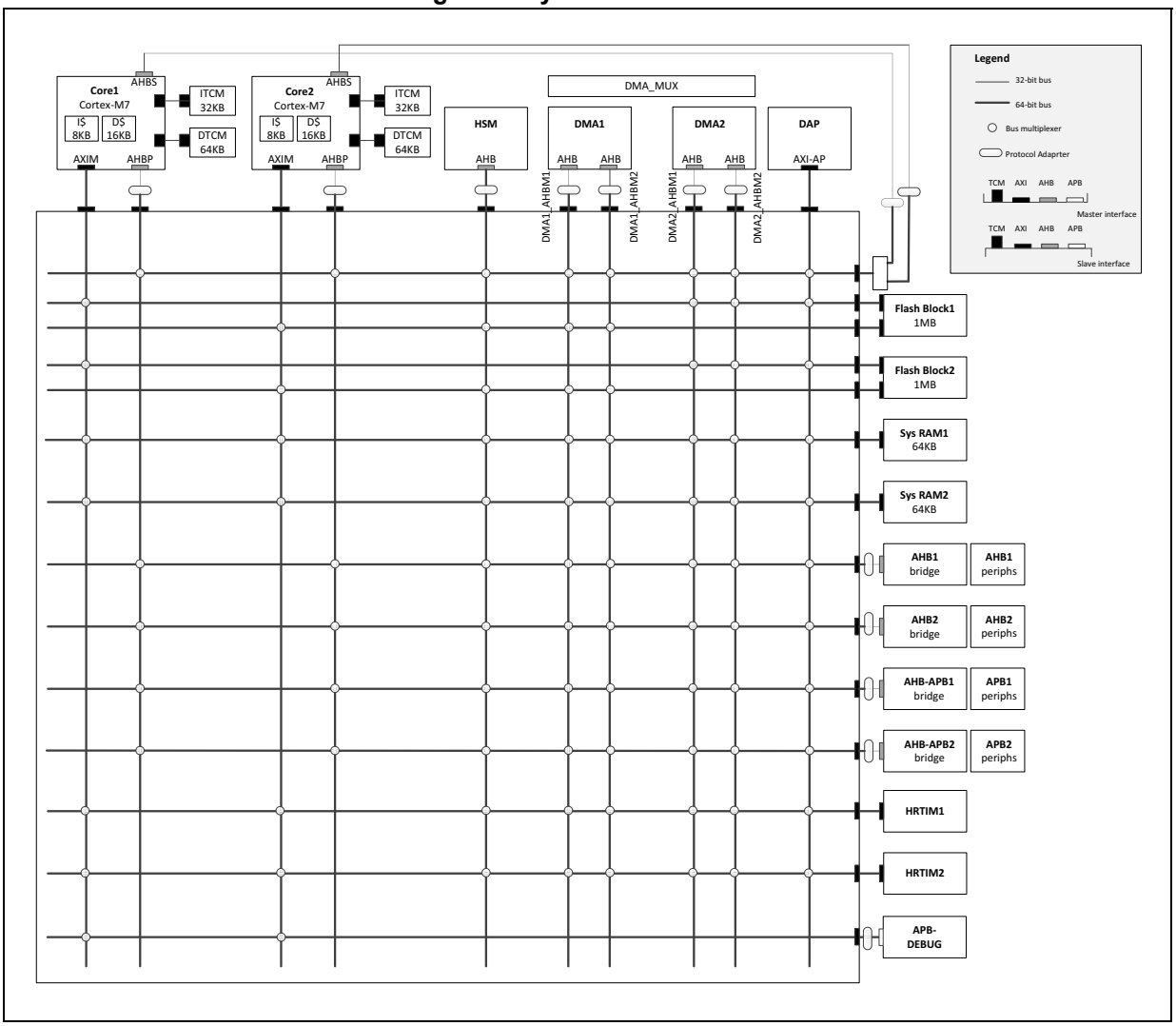
3.1 System architecture

The main system consists of an AXI bus matrix that interconnects:

- Up to 10 masters:
  - Cortex®-M7 core1 AXIM
  - Cortex®-M7 core1 AHB
  - Cortex®-M7 core2 AXIM
  - Cortex®-M7 core2 AHB
  - HSM
  - DMA1 AHB Master 1
  - DMA1 AHB Master 2
  - DMA2 AHB Master 1
  - DMA2 AHB Master 2
  - Trace and debug
- Up to 14 slaves:
  - Internal flash memory on the dual-ported flash Controller1
  - Internal flash memory on the dual-ported flash controller2
  - Internal SRAM1
  - Internal SRAM2
  - AHB1 peripherals
  - AHB2 peripherals
  - APB1 peripherals
  - APB2 peripherals
  - HRTIM1 on AXI interface
  - HRTIM2 on AXI interface
  - Core1 and Core2 caches and TCMs via Cortex®-M7 AHBS buses
  - Trace and debug

The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in *Figure 4*:

Figure 4. System architecture



3.1.1 TCM buses

The DTCM and ITCM (data and instruction tightly coupled RAMs) are connected through dedicated TCM buses directly to a Cortex®-M7 core. The DMA controllers can access the DTCM and ITCM through AHBS, a specific CPU slave AHB. An ITCMx is accessed by its related Cortex®-M7 corex at CPU clock speed, with zero wait states.

3.1.2 CPU buses

Cortex®-M7 AXIM bus

The Cortex®-M7 CPU uses the 64-bit AXIM bus to access all memories (excluding ITCM and DTCM) and HRTIMs peripherals.

The AXIM bus connects the CPU to the AXI bus matrix.

Cortex®-M7 ITCM bus

The Cortex®-M7 CPU uses the 64-bit ITCM bus for fetching instructions from and accessing data in the ITCM.

Cortex®-M7 DTCM bus

The Cortex®-M7 CPU uses the 2x32-bit DTCM bus for accessing data in the DTCM. The 2x32-bit DTCM bus allows load/load and load/store instruction pairs to be dual-issued on the DTCM memory. It can also fetch instructions.

Cortex®-M7 AHBS bus

The Cortex®-M7 CPU uses the 32-bit AHBS slave bus to allow the DMA controller, and other masters, to access the ITCM and the DTCM.

Cortex®-M7 AHBP bus

The Cortex®-M7 CPU uses the 32-bit AHBP bus for accessing AHB1, AHB2, APB1, and APB2 peripherals.

3.1.3 Bus master peripherals

DMA1 and DMA2 controllers

Each DMA controller (DMA1 and DMA2) has 32-bit master bus interfaces connected to AXI bus matrix.

The master bus interfaces allow DMA data transfers between two peripherals, between two memories or between a peripheral and a memory. Through the system bus matrices, the master bus interfaces can access all internal memories, including the ITCMs and DTCMs and all AHB and APB peripherals.

HSM

The HSM has one 32-bit bus interface connected to the AXI bus matrix.

3.1.4 TCM memories

Each ITCM and DTCM memory is accessible in two areas through different paths:

- Direct mode access, through the internal bus, providing a fast access to the core to its TCMs:
  - ITCM for instruction fetching in the range 0x0000_0000 - 0x0000_FFFF.
  - DTCM for data accesses in the range 0x2000_0000 - 0x2001_FFFF.
- Indirect mode access, through the interconnect matrix, providing access to external masters (other core and DMA modules). Refer to *Table 7*.

3.2 Memory organization

Table 4 and *Figure 5* show the device memory map for SR5E1x.

All addresses on the SR5E1x, including the reserved, are identified. The addresses represent the physical address assigned to each region or module. If a memory access,

either read or write, is attempted to any region marked as ‘Reserved’, then it results in a transfer error.

Table 4. Memory map overview

Start address	End address	Description
0x0000_0000	0x1FFF_FFFF	Cortex block0 code area, see *Table 5: Flash memory map*
0x2000_0000	0x3FFF_FFFF	Cortex block1 SRAM area, see *Table 6: RAM memory map*
0x4000_0000	0x5FFF_FFFF	Cortex block2 Peripherals area, see *Table 7: Peripheral memory map*
0x6000_0000	0x7FFF_FFFF	Cortex block 3 area, see *Table 8: CoreSight™ component memory map*
0x8000_0000	0x9FFF_FFFF	Reserved
0xA000_0000	0xBFFF_FFFF	Cortex block5 area, see *Table 9: AXI interface memory map*
0xC000_0000	0xDFFF_FFFF	Reserved
0xE000_0000	0xFFFF_FFFF	Cortex M7 internal peripherals

Table 5 is the detailed flash memory map for the SR5E1x.

Table 5. Flash memory map

Start address	End address	Description	Used size [Kbyte]
0x0000_0000	0x0000_7FFF	Alias region for ITCM (direct mode access)(1)	—
0x0000_8000	0x0002_7FFF	Alias region for HSM code(2)	—
0x0002_8000	0x07FF_FFFF	Reserved	—
Flash code 1			
0x0800_0000	0x0800_FFFF	Flash code 1 (low block B1F0)	64
0x0801_0000	0x0801_FFFF	Flash code 1 (low block B1F1)	64
0x0802_0000	0x0802_FFFF	Flash code 1 (low block B1F2)	64
0x0803_0000	0x0806_FFFF	Flash code 1 (A256 block B1F3)	256
0x0807_0000	0x080A_FFFF	Flash code 1 (A256 block B1F4)	256
0x080B_0000	0x080E_FFFF	Flash code 1 (A256 block B1F5)	256
Flash code 2			
0x080F_0000	0x080F_FFFF	Flash code 2 (low block B2F0)	64
0x0810_0000	0x0810_FFFF	Flash code 2 (low block B2F1)	64
0x0811_0000	0x0811_FFFF	Flash code 2 (low block B2F2)	64
0x0812_0000	0x0815_FFFF	Flash code 2 (A256 block B2F3)	256
0x0816_0000	0x0819_FFFF	Flash code 2 (A256 block B2F4)	256
0x081A_0000	0x081D_FFFF	Flash code 2 (A256 block B2F5)	256
0x081E_0000	0x08EF_FFFF	Reserved	—

Table 5. Flash memory map (continued)

Start address	End address	Description	Used size [Kbyte]
0x08F0_0000	0x08F0_3FFF	Data Flash (high block, D0_00)	16
0x08F0_4000	0x08F0_7FFF	Data Flash (high block, D0_01)	16
0x08F0_8000	0x08F0_BFFF	Data Flash (high block, D0_02)	16
0x08F0_C000	0x08F0_FFFF	Data Flash (high block, D0_03)	16
0x08F1_0000	0x17FF_FFFF	Reserved	—
0x1800_0000	0x1800_7FFF	HSM code flash (B0F0)	32
0x1800_8000	0x1801_7FFF	HSM code flash (B0F1)	64
0x1801_8000	0x1802_7FFF	HSM code flash (B0F2)	64
0x1802_8000	0x18EF_FFFF	Reserved	—
0x18F0_0000	0x18F0_3FFF	HSM data flash (mid block, D0_04)	16
0x18F0_4000	0x18F0_7FFF	HSM data flash (mid block, D0_05)	16
0x18F0_8000	0x1FEF_FFFF	Reserved	—
0x1FF0_0000	0x1FF0_3FFF	BAF	16
0x1FF0_4000	0x1FF7_FFFF	Reserved	—
0x1FF8_0000	0x1FF8_3FFF	UTest	16
0x1FF8_4000	0x1FFB_C27F	Reserved	—
0x1FFB_C280	0x1FFB_C2FF	Error locations for RWR1	—
0x1FFB_C300	0x1FFF_FFFF	Reserved	—

1. Used size of 32 Kbytes (local region) with direct access by M7 core in split mode. When Core1 and Core2 are in lockstep mode, the Core2 ITCM is connected to the Core1 ITCM interface and remapped to offer a continuous memory space of 64 Kbytes (local region: 0x0000_0000—0x0000_FFFF) of ITCM to Core1. Reserved for other masters.
2. Used size of 160 Kbytes (local region: 0x0000_0000--0x0002_7FFF) with direct access by M0+ core of HSM. Reserved for other masters. The HSM code is mapped and compiled in its local region and stored in the flash (B0F0, B0F1, B0F2). When Core1 and Core2 are in lockstep mode, no conflict for access by the HSM.

Table 6. RAM memory map

Start address	End address	Description	Used size [Kbyte]
0x2000_0000	0x2001_FFFF	Alias region for Cortex M7 DTCM (direct mode access)	64
0x2002_0000	0x23FF_FFFF	Reserved	—
0x2400_0000	0x2401_FFFF	SRAM1	128
0x2402_0000	0x2403_FFFF	SRAM2	128
0x2404_0000	0x3FFF_FFFF	Reserved	—

Table 7. Peripheral memory map

Start address	End address	Description	Used size [Kbyte]
0x4000_0000	0x4001_FFFF	APB1 peripherals, see *Table 10: APB1 peripheral mapping table*.	128
0x4002_0000	0x41FF_FFFF	Reserved	—
0x4200_0000	0x4201_FFFF	APB2 peripherals, see *Table 11: APB2 peripheral mapping table*.	128
0x4202_0000	0x43FF_FFFF	Reserved	—
0x4400_0000	0x4400_FFFF	AHB1 peripherals, see *Table 12: AHB1 peripheral mapping table*.	64
0x4401_0000	0x47FF_FFFF	Reserved	—
0x4800_0000	0x4800_FFFF	AHB2 peripherals, see *Table 13: AHB2 peripheral mapping table*.	64
0x4801_0000	0x59FF_FFFF	Reserved	—
0x5A00_0000	0x5A00_7FFF	Core1 ITCM (indirect mode access)	32
0x5A00_8000	0x5A00_FFFF	Reserved (split mode) Core1 ITCM extension (LS) (indirect mode access)	32
0x5A01_00	0x5A03_FFFF	Reserved	—
0x5A04_0000	0x5A04_7FFF	Core2 ITCM (split mode) (indirect mode access) Reserved (LS)	32
0x5A04_8000	0x5BFF_FFFF	Reserved	—
0x5C00_0000	0x5C00_FFFF	Core1 DTCM (indirect mode access)	64
0x5C01_0000	0x5C01_FFFF	Reserved (split mode) Core1 DTCM extension (LS) (indirect mode access)	64
0x5C02_0000	0x5C03_FFFF	Reserved	—
0x5C04_0000	0x5C04_FFFF	Reserved Core2 DTCM (split mode) (indirect mode access)	64
0x5C05_0000	0x5FFF_FFFF	Reserved	—

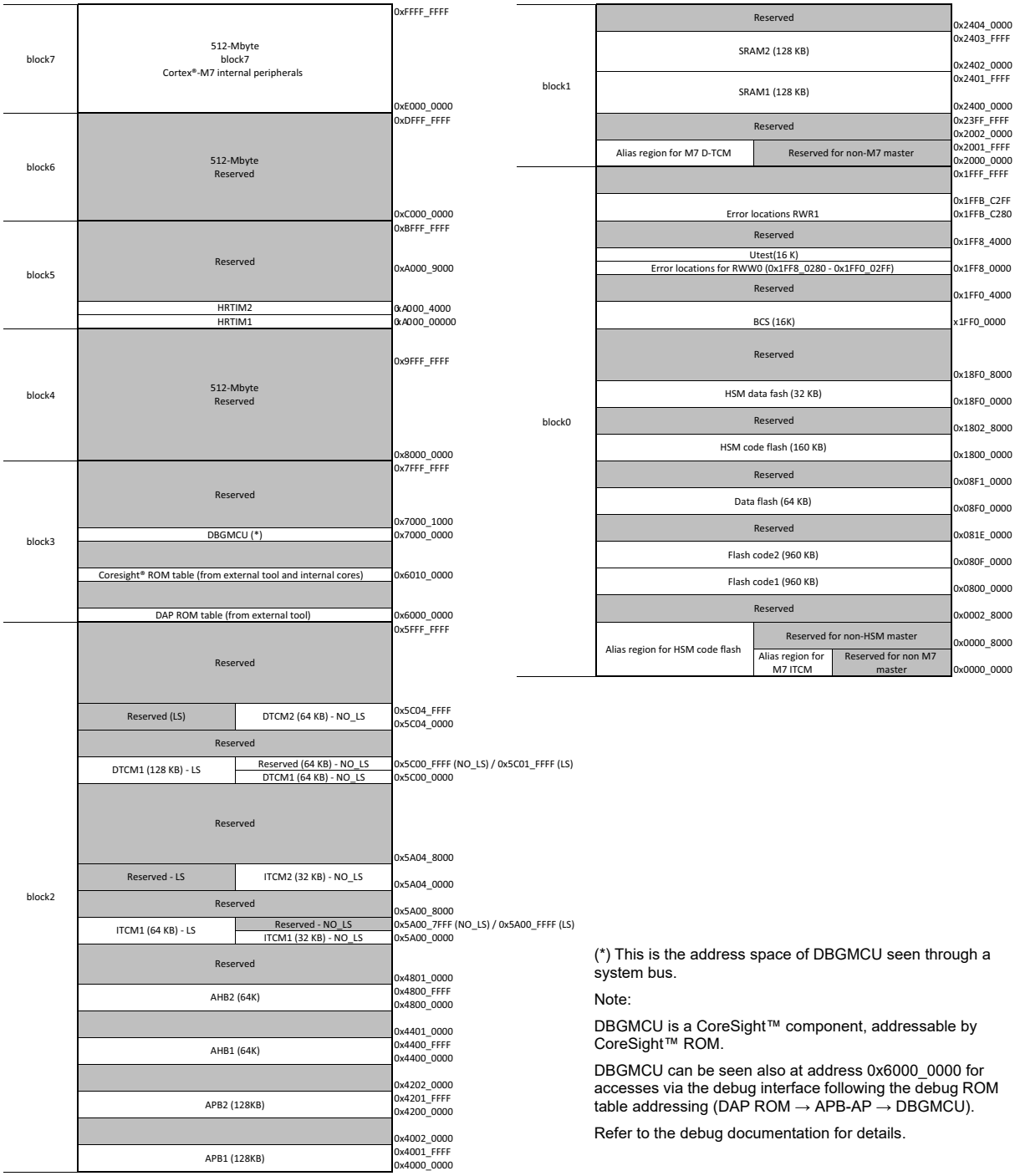
Table 8. CoreSight™ component memory map

Start address	End address	Description	Used size [Kbyte]
0x6000_0000	—	DAP ROM Table	—
0x6010_0000	—	CoreSight™ ROM Table	—
0x7000_0000	—	DBGMCU	—

Table 9. AXI interface memory map

Start address	End address	Description	Used size [Kbyte]
0xA000_0000	0xA000_3FFF	HRTIM1	—
0xA000_4000	0xA000_7FFF	HRTIM2	—

Figure 5. Memory mapping



(*) This is the address space of DBGMCU seen through a system bus.

Note:

DBGMCU is a CoreSight™ component, addressable by CoreSight™ ROM.

DBGMCU can be seen also at address 0x6000_0000 for accesses via the debug interface following the debug ROM table addressing (DAP ROM → APB-AP → DBGMCU).

Refer to the debug documentation for details.

Table 10. APB1 peripheral mapping table

Start address	End address	Index	Peripheral
0x4001 FC00	0x4001 FFFF	127	—
0x4001 F800	0x4001 FBFF	126	—
0x4001 F400	0x4001 F7FF	125	TDM
0x4001 F000	0x4001 F3FF	124	PASS
0x4001 EC00	0x4001 EFFF	123	SSCM
0x4001 E800	0x4001 EBFF	122	—
0x4001 E400	0x4001 E7FF	121	—
0x4001 E000	0x4001 E3FF	120	—
0x4001 DC00	0x4001 DFFF	119	—
0x4001 D800	0x4001 DBFF	118	PLL_DIG
0x4001 D400	0x4001 D7FF	117	—
0x4001 D000	0x4001 D3FF	116	PMU_DIG
0x4001 CC00	0x4001 CFFF	115	—
0x4001 C800	0x4001 CBFF	114	HSM2HOST itf (Host side)
0x4001 C400	0x4001 C7FF	113	—
0x4001 C000	0x4001 C3FF	112	AF Flash Reg interface
0x4001 BC00	0x4001 BFFF	111	—
0x4001 B800	0x4001 BBFF	110	—
0x4001 B400	0x4001 B7FF	109	—
0x4001 B000	0x4001 B3FF	108	—
0x4001 AC00	0x4001 AFFF	107	—
0x4001 A800	0x4001 ABFF	106	—
0x4001 A400	0x4001 A7FF	105	—
0x4001 A000	0x4001 A3FF	104	—
0x4001 9C00	0x4001 9FFF	103	—
0x4001 9800	0x4001 9BFF	102	—
0x4001 9400	0x4001 97FF	101	—
0x4001 9000	0x4001 93FF	100	—
0x4001 8C00	0x4001 8FFF	99	—
0x4001 8800	0x4001 8BFF	98	—
0x4001 8400	0x4001 87FF	97	—
0x4001 8000	0x4001 83FF	96	Flash Reg itf
0x4001 7C00	0x4001 7FFF	95	—
0x4001 7800	0x4001 7BFF	94	—
0x4001 7400	0x4001 77FF	93	—

Table 10. APB1 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4001 7000	0x4001 73FF	92	—
0x4001 6C00	0x4001 6FFF	91	—
0x4001 6800	0x4001 6BFF	90	—
0x4001 6400	0x4001 67FF	89	—
0x4001 6000	0x4001 63FF	88	—
0x4001 5C00	0x4001 5FFF	87	—
0x4001 5800	0x4001 5BFF	86	—
0x4001 5400	0x4001 57FF	85	—
0x4001 5000	0x4001 53FF	84	—
0x4001 4C00	0x4001 4FFF	83	—
0x4001 4800	0x4001 4BFF	82	—
0x4001 4400	0x4001 47FF	81	—
0x4001 4000	0x4001 43FF	80	STCU3
0x4001 3C00	0x4001 3FFF	79	
0x4001 3800	0x4001 3BFF	78	
0x4001 3400	0x4001 37FF	77	
0x4001 3000	0x4001 33FF	76	
0x4001 2C00	0x4001 2FFF	75	
0x4001 2800	0x4001 2BFF	74	
0x4001 2400	0x4001 27FF	73	
0x4001 2000	0x4001 23FF	72	
0x4001 1C00	0x4001 1FFF	71	
0x4001 1800	0x4001 1BFF	70	
0x4001 1400	0x4001 17FF	69	
0x4001 1000	0x4001 13FF	68	
0x4001 0C00	0x4001 0FFF	67	
0x4001 0800	0x4001 0BFF	66	
0x4001 0400	0x4001 07FF	65	
0x4001 0000	0x4001 03FF	64	
0x4000 FC00	0x4000 FFFF	63	FCCU
0x4000 F800	0x4000 FBFF	62	—
0x4000 F400	0x4000 F43F	61	CEM0
0x4000 F440	0x4000 F47F		CEM1
0x4000 F480	0x4000 F4BF		CEM2

Table 10. APB1 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4000 F4C0	0x4000 F4FF	61	CEM3
0x4000 F500	0x4000 F53F		CEM4
0x4000 F540	0x4000 F57F		CEM5
0x4000 F580	0x4000 F5BF		CEM6
0x4000 F5C0	0x4000 F5FF		CEM7
0x4000 F600	0x4000 F63F		CEM8
0x4000 F640	0x4000 F67F		CEM9
0x4000 F680	0x4000 F6BF		CEM10
0x4000 F6C0	0x4000 F6FF		CEM11
0x4000 F700	0x4000 F73F		CEM12
0x4000 F740	0x4000 F77F		CEM13
0x4000 F780	0x4000 F7BF		CEM14
0x4000 F000	0x4000 F3FF	60	IMA
0x4000 EC00	0x4000 EFFF	59	—
0x4000 E800	0x4000 EBFF	58	—
0x4000 E400	0x4000 E7FF	57	MEMU2
0x4000 E000	0x4000 E3FF	56	
0x4000 DC00	0x4000 DFFF	55	
0x4000 D800	0x4000 DBFF	54	
0x4000 D400	0x4000 D7FF	53	
0x4000 D000	0x4000 D3FF	52	
0x4000 CC00	0x4000 CFFF	51	
0x4000 C800	0x4000 CBFF	50	
0x4000 C400	0x4000 C7FF	49	
0x4000 C000	0x4000 C3FF	48	
0x4000 BC00	0x4000 BFFF	47	—
0x4000 B800	0x4000 BBFF	46	—
0x4000 B400	0x4000 B7FF	45	—
0x4000 B000	0x4000 B3FF	44	—
0x4000 AC00	0x4000 AFFF	43	—
0x4000 A800	0x4000 ABFF	42	—
0x4000 A400	0x4000 A7FF	41	—
0x4000 A000	0x4000 A3FF	40	—
0x4000 9C00	0x4000 9FFF	39	—
0x4000 9800	0x4000 9BFF	38	—

Table 10. APB1 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4000 9400	0x4000 97FF	37	—
0x4000 9000	0x4000 93FF	36	—
0x4000 8C00	0x4000 8FFF	35	—
0x4000 8800	0x4000 8BFF	34	—
0x4000 8400	0x4000 87FF	33	—
0x4000 8000	0x4000 83FF	32	—
0x4000 7C00	0x4000 7FFF	31	—
0x4000 7800	0x4000 7BFF	30	—
0x4000 7400	0x4000 77FF	29	—
0x4000 7000	0x4000 73FF	28	UART3
0x4000 6C00	0x4000 6FFF	27	UART2
0x4000 6800	0x4000 6BFF	26	—
0x4000 6400	0x4000 67FF	25	—
0x4000 6000	0x4000 63FF	24	SPI3
0x4000 5C00	0x4000 5FFF	23	SPI2
0x4000 5800	0x4000 5BFF	22	I2C2
0x4000 5400	0x4000 57FF	21	I2C1
0x4000 5000	0x4000 53FF	20	—
0x4000 4C00	0x4000 4FFF	19	TIM_TS
0x4000 4800	0x4000 4BFF	18	TIM7
0x4000 4400	0x4000 47FF	17	TIM6
0x4000 4000	0x4000 43FF	16	TIM3
0x4000 3C00	0x4000 3FFF	15	TIM2
0x4000 3800	0x4000 3BFF	14	—
0x4000 3400	0x4000 37FF	13	—
0x4000 3000	0x4000 33FF	12	—
0x4000 2C00	0x4000 2FFF	11	—
0x4000 2800	0x4000 2BFF	10	RTC
0x4000 2400	0x4000 27FF	9	—
0x4000 2000	0x4000 23FF	8	CMU1…CMU5
0x4000 1C00	0x4000 1FFF	7	—
0x4000 1800	0x4000 1BFF	6	—
0x4000 1400	0x4000 17FF	5	IWDG2
0x4000 1000	0x4000 13FF	4	IWDG1
0x4000 0C00	0x4000 0FFF	3	—

Table 10. APB1 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4000 0800	0x4000 0BFF	2	—
0x4000 0400	0x4000 07FF	1	WWDG2
0x4000 0000	0x4000 03FF	0	WWDG1

Table 11. APB2 peripheral mapping table

Start address	End address	Index	Peripheral
0x4201 FC00	0x4201 FFFF	127	—
0x4201 F800	0x4201 FBFF	126	—
0x4201 F400	0x4201 F7FF	125	—
0x4201 F000	0x4201 F3FF	124	—
0x4201 EC00	0x4201 EFFF	123	—
0x4201 E800	0x4201 EBFF	122	—
0x4201 E400	0x4201 E7FF	121	—
0x4201 E000	0x4201 E3FF	120	—
0x4201 DC00	0x4201 DFFF	119	—
0x4201 D800	0x4201 DBFF	118	—
0x4201 D400	0x4201 D7FF	117	—
0x4201 D000	0x4201 D3FF	116	—
0x4201 CC00	0x4201 CFFF	115	—
0x4201 C800	0x4201 CBFF	114	—
0x4201 C400	0x4201 C7FF	113	—
0x4201 C000	0x4201 C3FF	112	—
0x4201 BC00	0x4201 BFFF	111	—
0x4201 B800	0x4201 BBFF	110	—
0x4201 B400	0x4201 B7FF	109	—
0x4201 B000	0x4201 B3FF	108	—
0x4201 AC00	0x4201 AFFF	107	—
0x4201 A800	0x4201 ABFF	106	—
0x4201 A400	0x4201 A7FF	105	—
0x4201 A000	0x4201 A3FF	104	—
0x4201 9C00	0x4201 9FFF	103	—
0x4201 9800	0x4201 9BFF	102	—
0x4201 9400	0x4201 97FF	101	—
0x4201 9000	0x4201 93FF	100	—
0x4201 8C00	0x4201 8FFF	99	—

Table 11. APB2 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4201 8800	0x4201 8BFF	98	—
0x4201 8400	0x4201 87FF	97	—
0x4201 8000	0x4201 83FF	96	—
0x4201 7C00	0x4201 7FFF	95	—
0x4201 7800	0x4201 7BFF	94	—
0x4201 7400	0x4201 77FF	93	—
0x4201 7000	0x4201 73FF	92	—
0x4201 6C00	0x4201 6FFF	91	—
0x4201 6800	0x4201 6BFF	90	—
0x4201 6400	0x4201 67FF	89	—
0x4201 6000	0x4201 63FF	88	—
0x4201 5C00	0x4201 5FFF	87	—
0x4201 5800	0x4201 5BFF	86	—
0x4201 5400	0x4201 57FF	85	—
0x4201 5000	0x4201 53FF	84	—
0x4201 4C00	0x4201 4FFF	83	—
0x4201 4800	0x4201 4BFF	82	—
0x4201 4400	0x4201 47FF	81	—
0x4201 4000	0x4201 43FF	80	—
0x4201 3C00	0x4201 3FFF	79	—
0x4201 3800	0x4201 3BFF	78	—
0x4201 3400	0x4201 37FF	77	—
0x4201 3000	0x4201 33FF	76	—
0x4201 2C00	0x4201 2FFF	75	—
0x4201 2800	0x4201 2BFF	74	—
0x4201 2400	0x4201 27FF	73	—
0x4201 2000	0x4201 23FF	72	—
0x4201 1C00	0x4201 1FFF	71	—
0x4201 1800	0x4201 1BFF	70	—
0x4201 1400	0x4201 17FF	69	—
0x4201 1000	0x4201 13FF	68	—
0x4201 0C00	0x4201 0FFF	67	—
0x4201 0800	0x4201 0BFF	66	—
0x4201 0400	0x4201 07FF	65	—
0x4201 0000	0x4201 03FF	64	—

Table 11. APB2 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4200 FC00	0x4200 FFFF	63	FDCANs Message RAM (16K)
0x4200 F800	0x4200 FBFF	62	
0x4200 F400	0x4200 F7FF	61	
0x4200 F000	0x4200 F3FF	60	
0x4200 EC00	0x4200 EFFF	59	
0x4200 E800	0x4200 EBFF	58	
0x4200 E400	0x4200 E7FF	57	
0x4200 E000	0x4200 E3FF	56	
0x4200 DC00	0x4200 DFFF	55	
0x4200 D800	0x4200 DBFF	54	
0x4200 D400	0x4200 D7FF	53	
0x4200 D000	0x4200 D3FF	52	
0x4200 CC00	0x4200 CFFF	51	
0x4200 C800	0x4200 CBFF	50	
0x4200 C400	0x4200 C7FF	49	
0x4200 C000	0x4200 C3FF	48	
0x4200 BC00	0x4200 BFFF	47	FDCAN4
0x4200 B800	0x4200 BBFF	46	
0x4200 B400	0x4200 B7FF	45	FDCAN3
0x4200 B000	0x4200 B3FF	44	
0x4200 AC00	0x4200 AFFF	43	FDCAN2
0x4200 A800	0x4200 ABFF	42	
0x4200 A400	0x4200 A7FF	41	FDCAN1
0x4200 A000	0x4200 A3FF	40	
0x4200 9C00	0x4200 9FFF	39	—
0x4200 9800	0x4200 9BFF	38	—
0x4200 9400	0x4200 97FF	37	—
0x4200 9000	0x4200 93FF	36	—
0x4200 8C00	0x4200 8FFF	35	—
0x4200 8800	0x4200 8BFF	34	—
0x4200 8400	0x4200 87FF	33	SD_ADC2
0x4200 8000	0x4200 83FF	32	SD_ADC1
0x4200 7C00	0x4200 7FFF	31	—
0x4200 7800	0x4200 7BFF	30	—
0x4200 7400	0x4200 77FF	29	—

Table 11. APB2 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4200 7000	0x4200 73FF	28	—
0x4200 6C00	0x4200 6FFF	27	UART1
0x4200 6800	0x4200 6BFF	26	—
0x4200 6400	0x4200 67FF	25	—
0x4200 6000	0x4200 63FF	24	SPI4
0x4200 5C00	0x4200 5FFF	23	SPI1
0x4200 5800	0x4200 5BFF	22	—
0x4200 5400	0x4200 57FF	21	—
0x4200 5000	0x4200 53FF	20	—
0x4200 4C00	0x4200 4FFF	19	—
0x4200 4800	0x4200 4BFF	18	TIM16
0x4200 4400	0x4200 47FF	17	TIM15
0x4200 4000	0x4200 43FF	16	TIM5
0x4200 3C00	0x4200 3FFF	15	TIM4
0x4200 3800	0x4200 3BFF	14	—
0x4200 3400	0x4200 37FF	13	—
0x4200 3000	0x4200 33FF	12	TIM8_PWM
0x4200 2C00	0x4200 2FFF	11	TIM1_PWM
0x4200 2800	0x4200 2BFF	10	—
0x4200 2400	0x4200 27FF	9	—
0x4200 2000	0x4200 23FF	8	CMU0
0x4200 1C00	0x4200 1FFF	7	NVMPC2
0x4200 1800	0x4200 1BFF	6	NVMPC1
0x4200 1400	0x4200 17FF	5	PRAMC2
0x4200 1000	0x4200 13FF	4	PRAMC1
0x4200 0C00	0x4200 0FFF	3	SMPU
0x4200 0800	0x4200 0BFF	2	
0x4200 0400	0x4200 07FF	1	EXTI
0x4200 0000	0x4200 03FF	0	SYSCFG

Table 12. AHB1 peripheral mapping table

Start address	End address	Index	Peripheral
0x4400 FC00	0x4400 FFFF	63	—
0x4400 F800	0x4400 FBFF	62	—
0x4400 F400	0x4400 F7FF	61	—

Table 12. AHB1 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4400 F000	0x4400 F3FF	60	—
0x4400 EC00	0x4400 EFFF	59	—
0x4400 E800	0x4400 EBFF	58	—
0x4400 E400	0x4400 E7FF	57	—
0x4400 E000	0x4400 E3FF	56	—
0x4400 DC00	0x4400 DFFF	55	—
0x4400 D800	0x4400 DBFF	54	—
0x4400 D400	0x4400 D7FF	53	—
0x4400 D000	0x4400 D3FF	52	—
0x4400 CC00	0x4400 CFFF	51	—
0x4400 C800	0x4400 CBFF	50	—
0x4400 C400	0x4400 C7FF	49	—
0x4400 C000	0x4400 C3FF	48	—
0x4400 BC00	0x4400 BFFF	47	—
0x4400 B800	0x4400 BBFF	46	—
0x4400 B400	0x4400 B7FF	45	—
0x4400 B000	0x4400 B3FF	44	COMP2_dig (managing 4 compa ana)
0x4400 AC00	0x4400 AFFF	43	—
0x4400 A800	0x4400 ABFF	42	—
0x4400 A400	0x4400 A7FF	41	—
0x4400 A000	0x4400 A3FF	40	—
0x4400 9C00	0x4400 9FFF	39	—
0x4400 9800	0x4400 9BFF	38	DAC4 (CH1 + CH2) (Fast)
0x4400 9400	0x4400 97FF	37	DAC3 (CH1 + CH2) (Fast)
0x4400 9000	0x4400 93FF	36	—
0x4400 8C00	0x4400 8FFF	35	—
0x4400 8800	0x4400 8BFF	34	—
0x4400 8400	0x4400 87FF	33	—
0x4400 8000	0x4400 83FF	32	ADC3 - ADC4 - ADC5
0x4400 7C00	0x4400 7FFF	31	—
0x4400 7800	0x4400 7BFF	30	—
0x4400 7400	0x4400 77FF	29	—
0x4400 7000	0x4400 73FF	28	—
0x4400 6C00	0x4400 6FFF	27	—

Table 12. AHB1 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4400 6800	0x4400 6BFF	26	—
0x4400 6400	0x4400 67FF	25	—
0x4400 6000	0x4400 63FF	24	—
0x4400 5C00	0x4400 5FFF	23	—
0x4400 5800	0x4400 5BFF	22	—
0x4400 5400	0x4400 57FF	21	—
0x4400 5000	0x4400 53FF	20	—
0x4400 4C00	0x4400 4FFF	19	—
0x4400 4800	0x4400 4BFF	18	—
0x4400 4400	0x4400 47FF	17	—
0x4400 4000	0x4400 43FF	16	DMA2
0x4400 3C00	0x4400 3FFF	15	—
0x4400 3800	0x4400 3BFF	14	—
0x4400 3400	0x4400 37FF	13	CORDIC
0x4400 3000	0x4400 33FF	12	CRC
0x4400 2C00	0x4400 2FFF	11	HSEM
0x4400 2800	0x4400 2BFF	10	—
0x4400 2400	0x4400 27FF	9	—
0x4400 2000	0x4400 23FF	8	—
0x4400 1C00	0x4400 1FFF	7	—
0x4400 1800	0x4400 1BFF	6	—
0x4400 1400	0x4400 17FF	5	—
0x4400 1000	0x4400 13FF	4	—
0x4400 0C00	0x4400 0FFF	3	—
0x4400 0800	0x4400 0BFF	2	—
0x4400 0400	0x4400 07FF	1	—
0x4400 0000	0x4400 03FF	0	RCC

Table 13. AHB2 peripheral mapping table

Start address	End address	Index	Peripheral
0x4800 FC00	0x4800 FFFF	63	—
0x4800 F800	0x4800 FBFF	62	—
0x4800 F400	0x4800 F7FF	61	—
0x4800 F000	0x4800 F3FF	60	—
0x4800 EC00	0x4800 EFFF	59	—

Table 13. AHB2 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4800 E800	0x4800 EBFF	58	—
0x4800 E400	0x4800 E7FF	57	—
0x4800 E000	0x4800 E3FF	56	—
0x4800 DC00	0x4800 DFFF	55	—
0x4800 D800	0x4800 DBFF	54	—
0x4800 D400	0x4800 D7FF	53	—
0x4800 D000	0x4800 D3FF	52	—
0x4800 CC00	0x4800 CFFF	51	—
0x4800 C800	0x4800 CBFF	50	—
0x4800 C400	0x4800 C7FF	49	—
0x4800 C000	0x4800 C3FF	48	—
0x4800 BC00	0x4800 BFFF	47	—
0x4800 B800	0x4800 BBFF	46	—
0x4800 B400	0x4800 B7FF	45	—
0x4800 B000	0x4800 B3FF	44	COMP1_dig (managing 4 compa ana)
0x4800 AC00	0x4800 AFFF	43	—
0x4800 A800	0x4800 ABFF	42	—
0x4800 A400	0x4800 A7FF	41	—
0x4800 A000	0x4800 A3FF	40	—
0x4800 9C00	0x4800 9FFF	39	—
0x4800 9800	0x4800 9BFF	38	DAC2 (CH1 + CH2) (Fast)
0x4800 9400	0x4800 97FF	37	DAC1 (CH1 + CH2) (Fast)
0x4800 9000	0x4800 93FF	36	B-DAC1 (CH1 + CH2) (external)
0x4800 8C00	0x4800 8FFF	35	—
0x4800 8800	0x4800 8BFF	34	—
0x4800 8400	0x4800 87FF	33	—
0x4800 8000	0x4800 83FF	32	ADC1 - ADC2
0x4800 7C00	0x4800 7FFF	31	—
0x4800 7800	0x4800 7BFF	30	—
0x4800 7400	0x4800 77FF	29	—
0x4800 7000	0x4800 73FF	28	—
0x4800 6C00	0x4800 6FFF	27	—
0x4800 6800	0x4800 6BFF	26	—
0x4800 6400	0x4800 67FF	25	—

Table 13. AHB2 peripheral mapping table (continued)

Start address	End address	Index	Peripheral
0x4800 6000	0x4800 63FF	24	—
0x4800 5C00	0x4800 5FFF	23	—
0x4800 5800	0x4800 5BFF	22	—
0x4800 5400	0x4800 57FF	21	—
0x4800 5000	0x4800 53FF	20	—
0x4800 4C00	0x4800 4FFF	19	—
0x4800 4800	0x4800 4BFF	18	—
0x4800 4400	0x4800 47FF	17	DMAMUX1
0x4800 4000	0x4800 43FF	16	DMA1
0x4800 3C00	0x4800 3FFF	15	—
0x4800 3800	0x4800 3BFF	14	—
0x4800 3400	0x4800 37FF	13	—
0x4800 3000	0x4800 33FF	12	—
0x4800 2C00	0x4800 2FFF	11	—
0x4800 2800	0x4800 2BFF	10	—
0x4800 2400	0x4800 27FF	9	—
0x4800 2000	0x4800 23FF	8	GPIOI
0x4800 1C00	0x4800 1FFF	7	GPIOH
0x4800 1800	0x4800 1BFF	6	GPIOG
0x4800 1400	0x4800 17FF	5	GPIOF
0x4800 1000	0x4800 13FF	4	GPIOE
0x4800 0C00	0x4800 0FFF	3	GPIOD
0x4800 0800	0x4800 0BFF	2	GPIOC
0x4800 0400	0x4800 07FF	1	GPIOB
0x4800 0000	0x4800 03FF	0	GPIOA

3.3 Embedded SRAM

The SR5E1x device features:

- 256 Kbytes of system SRAM
- 128 Kbytes of data TCM RAM
- 64 Kbytes of instruction TCM RAM

The embedded system RAM is divided in two blocks:

- AXI SRAM1, 128 Kbytes is mapped at the address 0x2400_0000 and accessible by all system masters.
- AXI SRAM2, 128 Kbytes is mapped at the address 0x2402_0000 and accessible by all system masters.

The system AXI SRAM can be accessed as bytes, half-words, words, or doublewords (64-bit units). These memories can be addressed at maximum system clock frequency without wait state.

The TCM SRAMs are dedicated to the Cortex®-M7 cores:

- DTCM1 on the Core1 TCM interface is mapped at the address 0x5C00_0000 and accessible by Cortex®-M7 Core1, and by DMAs or Core2 through the AHBS slave bus of the Cortex®-M7 Core1.
- DTCM2 on the Core2 TCM interface is mapped at the address 0x5C04_0000 and accessible by Cortex®-M7 Core2, and by DMAs or Core1 through the AHBS slave bus of the Cortex®-M7 Core2.

The DTCMs can be used as a read-write segment to host critical real-time data (such as stack and heap) for an application running on a Cortex®-M7 core.

- ITCM1 on Core1 TCM interface is mapped at the address 0x5A00_0000 and accessible by Cortex®-M7 Core1 and by the DMAs or the Core2 through the AHBS slave bus of the Cortex®-M7 Core1.
- ITCM2 on Core2 TCM interface is mapped at the address 0x5A04_0000 and accessible by Cortex®-M7 Core2 and by the DMAs or the Core1 through the AHBS slave bus of the Cortex®-M7 Core2.

The ITCMs can be used to host code for time-critical routines (such as interrupt handlers) that requires deterministic execution.

Note: When Core1 and Core2 are in lock-step mode, ITCM2 and DTCM2 are connected to Core1 TCM interfaces and remapped to offer a continuous range of 128 Kbytes of DTCM and 64 Kbytes of ITCM to Core1.

3.4 Flash memory overview

Throughout the document, both flash and nonvolatile memory (NVM) are used indistinctly because SR5E1x uses flash memory as a type of NVM. Note that for flash memory controller PFLASHC and NVMC are used indistinctly in the document as well. The flash memory controllers (NVMPC1, NVMPC2) manage CPU AXI accesses to the flash memory. They implement the erase and program flash memory operations and the read and write protection mechanisms.

The flash memory is organized as follows:

- Two main memory blocks divided into sectors.
- An information block:
  - System memory from which the device boots in system memory boot mode.
  - Option bytes to configure read and write protection.

Refer to *Chapter 17: Embedded flash memory* for more details.

4 Signal description

4.1 Production packages

SR5E1x production packages (case number and outline drawings) are provided in the SR5E1x Microcontroller datasheet.

4.2 Package pinouts and ballouts

The SR5E1x pin package pinouts and ballouts are contained in the Microsoft Excel® workbook file attached to the SR5E1x IO_Definition document.

5 Device configuration

5.1 Core modules

The SR5E1x has three cores in two distinct modules:

- A Cortex®-M0+ in the hardware security module (HSM), refer to SR5E1x security reference manual.
- Two Cortex®-M7 in the main platform.

The two Cortex®-M7, aka Core1 and Core2, can be configured:

- In decoupled mode, offering two processing units.
- In lock-step mode, offering one processing unit.

The two Cortex®-M7 have the following features:

- Single and double precision FPU.
- 16 regions of Core Memory Protection Unit (CMPU).

5.2 System modules

5.2.1 Interconnect / bus matrix

The SR5E1x device contains one bus matrix, based on the NIC400. This bus matrix, or interconnect, ensures and arbitrates concurrent accesses from multiple masters to multiple slaves. This allows efficient and simultaneous operation from cores and DMAs to memories and peripherals.

The arbitration uses a round-robin algorithm with Quality of Service (QoS) capability (least recently used (LRU) priority).

The interconnect has 9 functional initiator ports (INIx) and one debug master port. These ports are also referred to as ASIB (AMBA® Slave Interface Block).

The interconnect has 13 functional target ports (TARGx) and one debug slave port. These ports are also referred to as AMIB (AMBA® Master Interface Block).

5.2.1.1 Quality of service

Traffic priority management (QoS - Quality of Service) is configured at start-up via device configuration format (DCF) records, allowing the user to assign the right level of priority to each initiator both in read and write.

The AXI switch matrix uses a priority-based arbitration when two initiators simultaneously attempt to access the same target. Each initiator has programmable read channel and write channel priorities, known as QoS, from 0 to 15, such that the higher the value, the higher the priority.

The read channel QoS value is programmed in the AXI interconnect by INIx Read QoS DCFs (AXI_INIx_READ_QOS_DCFy), and the write channel in the AXI interconnect by INIx Write QoS DCFs (AXI_INIx_WRITE_QOS_DCFy).

Refer to *Chapter 7: Device configuration format (DCF) records* for detailed information on the DCFs.

The following table shows the default QoS configuration after reset, when no DCFs are programmed.

Table 14. Default QoS R/W value

Initiator port (INIx)	QoS value
Core1 - AXIM	0000
Core1 - AHBP	0000
Core2 - AXIM	0000
Core2 - AHBP	0000
DMA1 - AHBMEM	0001
DMA1 - AHBPER	0001
DMA2 - AHBMEM	0001
DMA2 - AHBPER	0001
HSM	0010

If two coincident transactions arrive at the same target, the higher priority transaction passes before the lower priority. If the two transactions have the same QoS value, then a least recently-used (LRU) priority scheme is adopted.

The QoS values are programmed according to the latency requirements for the application. Setting a higher priority for an Initiator ensures a lower latency for transactions initiated by the associated bus master.

5.2.1.2 INIx read/write QoS DCFs description

AXI_INIx_READ_QOS_DCF0 INIx Read QoS DCF0 description

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
HSM_AR_QoS	Reserved	CORE2_AHBP_AR_QoS	CORE2_AXIM_AR_QoS	CORE1_AHBP_AR_QoS	CORE1_AXIM_AR_QoS

Reset: 0x0000_1111

31:28 **HSM_AR_QoS**
Read channel QoS setting for HSM
0x0: Lowest priority
0xF: Highest priority

27:16 Reserved

15:12 **CORE2_AHBP_AR_QoS**
Read channel QoS setting for Core2 AHBP
0x0: Lowest priority
0xF: Highest priority

11:8 **CORE2_AXIM_AR_QoS**
Read channel QoS setting for Core2 AXIM
0x0: Lowest priority
0xF: Highest priority

7:4 **CORE1_AHBP_AR_QoS**
Read channel QoS setting for Core1 AHBP
0x0: Lowest priority
0xF: Highest priority

3:0 **CORE1_AXIM_AR_QoS**
Read channel QoS setting for Core1 AXIM
0x0: Lowest priority
0xF: Highest priority

AXI_INIx_READ_QOS_DCF1 INIx Read QoS DCF1 description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DMA2_AHBPER_AR_QoS				DMA2_AHBMEM_AR_QoS				DMA1_AHBPER_AR_QoS				DMA1_AHBMEM_AR_QoS			

Reset: 0x0000_1111

31:16 Reserved

15:12 **DMA2_AHBPER_AR_QoS**
Read channel QoS setting for DMA2 AHBPER
0x0: Lowest priority
0xF: Highest priority

11:8 **DMA2_AHBMEM_AR_QoS**
Read channel QoS setting for DMA2 AHBMEM
0x0: Lowest priority
0xF: Highest priority

7:4 **DMA1_AHBPER_AR_QoS**
Read channel QoS setting for DMA1 AHBPER
0x0: Lowest priority
0xF: Highest priority

3:0 **DMA1_AHBMEM_AR_QoS**
Read channel QoS setting for DAM1 AHBMEM
0x0: Lowest priority
0xF: Highest priority

AXI_INIx_WRITE_QOS_DCF0 INIx Write QoS DCF0 description

31 30 29 28	27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
HSM_AW_QoS	Reserved	CORE2_AHBP_Aw_QoS	CORE2_AXIIM_AW_QoS	CORE1_AHBP_AW_QoS	CORE1_AXIIM_AW_QoS

Reset: 0x2000_0000

31:28 **HSM_AW_QoS**
Write channel QoS setting for HSM
0x0: Lowest priority
0xF: Highest priority

27:16 Reserved

15:12 **CORE2_AHBP_AW_QoS**
Write channel QoS setting for Core2 AHBP
0x0: Lowest priority
0xF: Highest priority

11:8 **CORE2_AXIM_AW_QoS**
Write channel QoS setting for Core2 AXIM
0x0: Lowest priority
0xF: Highest priority

7:4 **CORE1_AHBP_AW_QoS**
Write channel QoS setting for Core1 AHBP
0x0: Lowest priority
0xF: Highest priority

3:0 **CORE1_AXIM_AW_QoS**
Write channel QoS setting for Core1 AXIM
0x0: Lowest priority
0xF: Highest priority

AXI_INIx_WRITE_QOS_DCF1 INIx Write QoS DCFs1description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DMA2_AHBPER_AW_QoS				DMA2_AHBMEM_AW_QoS				DMA1_AHBPER_AW_QoS				DMA1_AHBMEM_AW_QoS			

Reset: 0x0000_1111

31:16 Reserved

15:12 **DMA2_AHBPER_AW_QoS**
Write channel QoS setting for DMA2 AHBPER
0x0: Lowest priority
0xF: Highest priority

11:8 **DMA2_AHBMEM_AW_QoS**
Write channel QoS setting for DMA2 AHBMEM
0x0: Lowest priority
0xF: Highest priority

7:4 **DMA1_AHBPER_AW_QoS**
Write channel QoS setting for DMA1 AHBPER
0x0: Lowest priority
0xF: Highest priority

3:0 **DMA1_AHBMEM_AW_QoS**
Write channel QoS setting for DAM1 AHBMEM
0x0: Lowest priority
0xF: Highest priority

AXI_WDG_DCF AXI Transactions Watchdog DCF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDG_CNT																							

Reset: 0x00FF_FFFF

31:24 Reserved

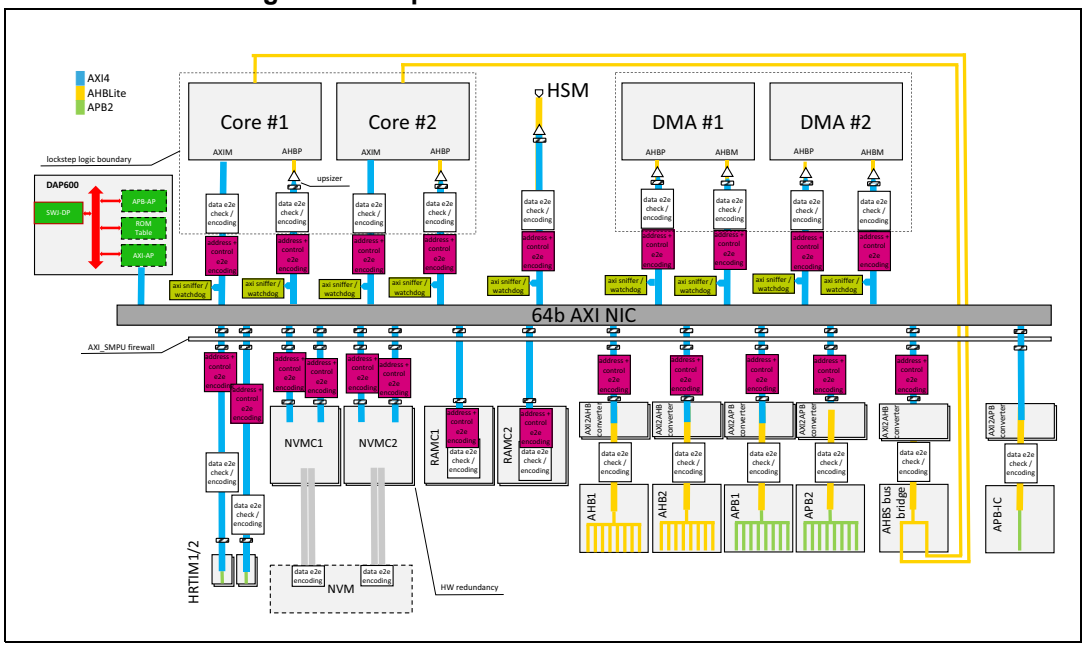
23:0 **WDG_CNT**
Watchdog Counter Value

5.2.1.3 NIC and E2EECC protection

The connections between NIC masters and slaves (clients) are denoted as data paths. Data corruption on all data paths between the Safety Masters and any client is detected with at least 99% coverage via two main safety mechanisms: data from the replicated cores is encoded using Error Correcting Code (ECC), which is implemented with a single-error correction, double-error detection (SECDED) code with a Hamming distance of 4 and includes coverage of addressing information; ECC is also applied to control signals and address decoding, to verify that the data reaches all of the intended clients, from all possible connections to these clients and the intended operation is performed on the target address. The transactions are monitored by an AXI sniffer and watchdog, to avoid system stalls.

The following figure shows a general view of ECC schema and NIC protection.

Figure 6. NIC protection and E2EECC schematic



ECC bits are generated on writes by NIC masters (including, but not limited to the Safety Core) and checked on reads. The ECC correction bits are stored alongside the data in NVM and RAM so, in principle, no ECC logic is necessary at the memories themselves. For this reason, the ECC schema is referred to as End-to-End ECC (E2E ECC) in the following sections. For NIC master and slaves, apart from memories, new ECC logic is added as

these clients cannot store or produce the ECC correction bits. This resolves the problem where ECC needs to be calculated in real time before entering or exiting the ECC-protected data path. This setup leaves the data path downstream of the ECC units (for example, starting at, and downstream of, the peripheral bridges) unprotected by ECC. To detect corruptions introduced in those unprotected data paths, the BRIDGEs and peripherals can be used redundantly, or other mechanisms may be used to validate peripheral operation.

The E2E ECC schema provides high detection capabilities against failures affecting the data content of the transaction. The inclusion of the target address in the computation of the redundancy bits (8 ECC bits) does allow the partial detection of addressing faults as well. To reach the desired integrity level, additional dedicated safety mechanisms are implemented in the data path to:

- Improve the detection capability over addressing failures (no/multiple/wrong address selected), considering faults affecting address transmission (from master to client) as well as the decoding of the address.
- Provide coverage for control failures affecting, for example, the type (read vs. write) or size of a transaction.

Though the safety mechanisms protecting the NIC, the RAM controller, or the Flash memory controller are different, they are all based on the feedback of address and control information from the target to the source of the transaction, which is responsible for checking for consistency with respect to the intended transaction. Depending on the portion of the data path covered by the specific safety mechanism, the source can be an NIC master port rather than the NIC interface of the RAM or Flash controllers; the target is respectively an NIC slave port, the RAM array, or the Flash module (refer to “NIC”, “Non-Volatile Memory Platform Controller (NVMPC)”, and “Platform RAM controller AXI (PRAMC_AXI)” chapters).

5.2.2 Hardware semaphore (HSEM2) configuration

The peripheral platform includes a memory-mapped semaphore module (HSEM2) that provides:

- Robust hardware support needed in multicore systems for implementing semaphores
- A simple mechanism to achieve “lock/unlock” operations via a single write access

The hardware semaphore block provides 32 (32-bit) registers based semaphores.

The semaphores can be used to ensure synchronization between different processes running between different cores.

5.2.3 System memory protection unit (SMPU) configuration

The SR5E1x contains one instance of the System Memory Protection Unit (SMPU) with 24 regions.

This section summarizes how the module instance has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

Table 15. Master ID assignments

SMPU logical bus master number	Bus master
0’b1000	Core1 (Cortex®-M7)
0’b0001	Core2 (Cortex®-M7)

Table 15. Master ID assignments (continued)

SMPU logical bus master number	Bus master
0’b0010	HSM
0’b1011	DMA1
0’b0100	DMA2
0’b1101	Debug Access Port (DAP)

5.2.4 Nested vectored interrupt controller configuration

Both CPU1 (Cortex®-M7) and CPU2 (Cortex®-M7) cores have their own nested vector interrupt controller (respectively NVIC1 and NVIC2).

Each CPU has its own exceptions connected to its Nested vectored interrupt controller (NVIC): reset, NMI, HardFault, MemManage, Bus Fault, UsageFault, SVCall, DebugMonitor, PendSV, SysTick.

Each CPU is able to generate an interrupt or trigger a wake-up to the other core through the EXTI. Cortex®-M7 core features an output signal as a result of the Send Event (SEV) instruction. The Core1_SEV is connected to Core2 NVIC (NVIC2) position 151, while Core2_SEV is connected to Core1 NVIC (NVIC1) position 150.

Each CPU also has its own Floating Point Unit (FPU) interrupt connected to its Nested Vectored Interrupt Controller (NVIC), in position 81.

The Window Watchdog 1 (WWDG1) interrupt is connected to the CPU1 NVIC (NVIC1) position 0, while Window Watchdog 2 (WWDG2) interrupt is connected to the CPU2 NVIC (NVIC2) position 0.

5.2.4.1 Interrupt and exception vectors

The gray rows in the following table describe the vectors without specific position.

Table 16. SR5E1x vector table

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
—	—	—	—	Reserved	0x0000 0000
Fixed	—	—	Reset	Reset	0x0000 0004
Fixed	—	—	NMI	FCCU Non maskable interrupt	0x0000 0008
Fixed	—	—	HardFault	All class of fault	0x0000 000C
Settable	—	—	MemManage	Memory management	0x0000 0010
Settable	—	—	BusFault	Pre-fetch fault; memory access fault	0x0000 0014
Settable	—	—	UsageFault	Undefined instruction or illegal state	0x0000 0018

Table 16. SR5E1x vector table (continued)

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
—	—	—	—	Reserved	0x0000 001C-0x0000 002B
Settable	—	—	SVCall	System Service call via SWI instruction	0x0000 002C
Settable	—	—	Debug Monitor	Debug Monitor	0x0000 0030
—	—	—	—	Reserved	0x0000 0034
Settable	—	—	PendSV	Pendable request for system service	0x0000 0038
Settable	—	—	Systick	System tick timer	0x0000 003C
Settable	0	—	WWDG1	Window Watchdog Early interrupt	0x0000 0040
—	—	0	WWDG2	Window Watchdog Early interrupt	0x0000 0040
Settable	1	1	RTC_ALARM	RTC Alarm interrupts through EXTI line 18	0x0000 0044
Settable	2	2	—	Reserved	0x0000 0048
Settable	3	3	RTC_WKPU	RTC_Wakeup through EXTI line 20 interrupt	0x0000 004C
Settable	4	4	FLASH	Flash global interrupt	0x0000 0050
Settable	5	5	RCC	RCC global interrupt	0x0000 0054
Settable	6	6	EXTI0	EXTI Line0 interrupt	0x0000 0058
Settable	7	7	EXTI1	EXTI Line1 interrupt	0x0000 005C
Settable	8	8	EXTI2	EXTI Line2 interrupt	0x0000 0060
Settable	9	9	EXTI3	EXTI Line3 interrupt	0x0000 0064
Settable	10	10	EXTI4	EXTI Line4 interrupt	0x0000 0068
Settable	11	11	DMA1_Stream0	DMA1 Stream0 global interrupt	0x0000 006C
Settable	12	12	DMA1_Stream1	DMA1 Stream1 global interrupt	0x0000 0070
Settable	13	13	DMA1_Stream2	DMA1 Stream2 global interrupt	0x0000 0074
Settable	14	14	DMA1_Stream3	DMA1 Stream3 global interrupt	0x0000 0078
Settable	15	15	DMA1_Stream4	DMA1 Stream4 global interrupt	0x0000 007C
Settable	16	16	DMA1_Stream5	DMA1 Stream5 global interrupt	0x0000 0080
Settable	17	17	DMA1_Stream6	DMA1 Stream6 global interrupt	0x0000 0084
Settable	18	18	DMA1_Stream7	DMA1 Stream7 global interrupt	0x0000 0088
Settable	19	19	TIM15	TIM15 global interrupt	0x0000 008C
Settable	20	20	TIM16	TIM16 global interrupt	0x0000 0090
Settable	21	21	—	Reserved	0x0000 0094
Settable	22	22	—	Reserved	0x0000 0098

Table 16. SR5E1x vector table (continued)

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
Settable	23	23	—	Reserved	0x0000 009C
Settable	24	24	TIM1_BRK/TIM1_TERR/TIM1_IERR	TIM1 Break interrupt/TIM1 Transition/TIM1 Index Error	0x0000 00A0
Settable	25	25	TIM1_UP	TIM1 Update interrupt	0x0000 00A4
Settable	26	26	TIM1_TRG_COM/TIM1_DIR/TIM1_IDX	TIM1 Trigger and Commutation interrupts/TIM1 Direction Change/TIM1_Index	0x0000 00A8
Settable	27	27	TIM1_CC	TIM1 Capture Compare interrupt (OR of all channels)	0x0000 00AC
Settable	28	28	TIM2	TIM2 global interrupt	0x0000 00B0
Settable	29	29	TIM3	TIM3 global interrupt	0x0000 00B4
Settable	30	30	TIM4	TIM4 global interrupt	0x0000 00B8
Settable	31	31	I2C1_EV	I2C1 event interrupt OR EXTI line 23 interrupt	0x0000 00BC
Settable	32	32	I2C1_ER	I2C1 error interrupt	0x0000 00C0
Settable	33	33	I2C2_EV	I2C2 event interrupt OR EXTI line 24 interrupt	0x0000 00C4
Settable	34	34	I2C2_ER	I2C2 error interrupt	0x0000 00C8
Settable	35	35	SPI1	SPI1 global interrupt	0x0000 00CC
Settable	36	36	SPI2	SPI2 global interrupt	0x0000 00D0
Settable	37	37	UART1	UART1 global interrupt OR EXTI line 25 interrupt	0x0000 00D4
Settable	38	38	UART2	UART2 global interrupt OR EXTI line 26 interrupt	0x0000 00D8
Settable	39	39	UART3	UART3 global interrupt OR EXTI line 28 interrupt	0x0000 00DC
Settable	40	40	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 00E0
Settable	41	41	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E4
Settable	42	42	—	Reserved	0x0000 00E8
Settable	43	43	TIM8_BRK/TIM8_TERR/TIM8_IERR	TIM8 Break interrupt/Tim8 Transition/TIM8 Index Error	0x0000 00EC
Settable	44	44	TIM8_UP	TIM8 Update interrupt	0x0000 00F0
Settable	45	45	TIM8_TRG_COM/TIM8_DIR/TIM8_IDX	TIM8 Trigger and Commutation interrupt/TIM8 Direction Change Interrupt/TIM8 Index	0x0000 00F4
Settable	46	46	TIM8_CC	TIM8 Capture Compare interrupt (OR of all channels)	0x0000 00F8
Settable	47	47	TIM5	TIM5 global interrupt	0x0000 00FC
Settable	48	48	TIM6	TIM6 global interrupt	0x0000 0100
Settable	49	49	TIM7	TIM7 global interrupt	0x0000 0104

Table 16. SR5E1x vector table (continued)

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
Settable	50	50	TIM_TS	TIMTS_global interrupt	0x0000 0108
Settable	51	51	SPI3	SPI3 global interrupt	0x0000 010C
Settable	52	52	SPI4	SPI4 global interrupt	0x0000 0110
Settable	53	53	—	Reserved	0x0000 0114
Settable	54	54	—	Reserved	0x0000 0118
Settable	55	55	—	Reserved	0x0000 011C
Settable	56	56	DMA2_Stream0	DMA2 Stream0 global interrupt	0x0000 0120
Settable	57	57	DMA2_Stream1	DMA2 Stream1 global interrupt	0x0000 0124
Settable	58	58	DMA2_Stream2	DMA2 Stream2 global interrupt	0x0000 0128
Settable	59	59	DMA2_Stream3	DMA2 Stream3 global interrupt	0x0000 012C
Settable	60	60	DMA2_Stream4	DMA2 Stream4 global interrupt	0x0000 0130
Settable	61	61	DMA2_Stream5	DMA2 Stream5 global interrupt	0x0000 0134
Settable	62	62	DMA2_Stream6	DMA2 Stream6 global interrupt	0x0000 0138
Settable	63	63	DMA2_Stream7	DMA2 Stream7 global interrupt	0x0000 013C
Settable	64	64	DMAMUX_OVR	DMAMUX Overrun interrupt	0x0000 0140
Settable	65	65	—	Reserved	0x0000 0144
Settable	66	66	—	Reserved	0x0000 0148
Settable	67	67	HRTIM1_Master_Int	HRTIM1 Master Timer Interrupt (hrtim_it1)	0x0000 014C
Settable	68	68	HRTIM1_TIMA_Int	HRTIM1 Timer A Interrupt (hrtim_it2)	0x0000 0150
Settable	69	69	HRTIM1_TIMB_Int	HRTIM1 Timer B Interrupt (hrtim_it3)	0x0000 0154
Settable	70	70	HRTIM1_TIMC_Int	HRTIM1 Timer C Interrupt (hrtim_it4)	0x0000 0158
Settable	71	71	HRTIM1_TIMD_Int	HRTIM1 Timer D Interrupt (hrtim_it5)	0x0000 015C
Settable	72	72	HRTIM1_TIME_Int	HRTIM1 Timer E Interrupt (hrtim_it6)	0x0000 0160
Settable	73	73	HRTIM1_TIM_FLT_Int	HRTIM1 Fault Interrupt (hrtim_it8)	0x0000 0164
Settable	74	74	HRTIM1_TIMF_Int	HRTIM1 Timer F Interrupt (hrtim_it7)	0x0000 0168
Settable	75	75	—	Reserved	0x0000 016C
Settable	76	76	—	Reserved	0x0000 0170
Settable	77	77	—	Reserved	0x0000 0174
Settable	78	78	—	Reserved	0x0000 0178
Settable	79	79	—	Reserved	0x0000 017C
Settable	80	80	—	Reserved	0x0000 0180

Table 16. SR5E1x vector table (continued)

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
Settable	81	—	FPU1	CPU1 FPU (FPIDC1 and FPU1_IT_EN(4)) or (FPOFC1 and FPU1_IT_EN(3)) or (FPUFC1 and FPU1_IT_EN(2)) or (FPDZC1 and FPU1_IT_EN(1)) or (FPIOC1 and FPU1_IT_EN(0));	0x0000 0184
Settable	—	81	FPU2	CPU2 FPU ((FPIDC2 and FPU2_IT_EN(4)) or (FPOFC2 and FPU2_IT_EN(3)) or (FPUFC2 and FPU2_IT_EN(2)) or (FPDZC2 and FPU2_IT_EN(1)) or (FPIOC2 and FPU2_IT_EN(0));	0x0000 0184
Settable	82	82	—	Reserved	0x0000 0188
Settable	83	83	—	Reserved	0x0000 018C
Settable	84	84	m_can_1_int0	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_1_IR[0] & ~M_CAN_1_ILS[0])| ... | (M_CAN_1_IR[11] & ~M_CAN_1_ILS[11]) | (M_CAN_1_IR[13] & ~M_CAN_1_ILS[13]) | … | (M_CAN_1_IR[31] & ~M_CAN_1_ILS[31])	0x0000 0190
Settable	85	85	m_can_1_int1	Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_1_IR[0] & M_CAN_1_ILS[0])| ... | (M_CAN_1_IR[11] & M_CAN_1_ILS[11]) | (M_CAN_1_IR[13] & M_CAN_1_ILS[13]) | … | (M_CAN_1_IR[31] & M_CAN_1_ILS[31])	0x0000 0194
Settable	86	86	mcan_1_ dmu_int	MCAN1 DMU Interrupt	0x0000 0198

Table 16. SR5E1x vector table (continued)

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
Settable	87	87	m_can_2_int0	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_2_IR[0] & ~M_CAN_2_ILS[0])| ... | (M_CAN_2_IR[11] & ~M_CAN_2_ILS[11]) | (M_CAN_2_IR[13] & ~M_CAN_2_ILS[13]) | … | (M_CAN_2_IR[31] & ~M_CAN_2_ILS[31])	0x0000 019C
Settable	88	88	m_can_2_int1	Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_2_IR[0] & M_CAN_2_ILS[0])| ... | (M_CAN_2_IR[11] & M_CAN_2_ILS[11]) | (M_CAN_2_IR[13] & M_CAN_2_ILS[13]) | … | (M_CAN_2_IR[31] & M_CAN_2_ILS[31])	0x0000 01A0
Settable	89	89	mcan_2_ dmu_int	MCAN2 DMU Interrupt	0x0000 01A4
Settable	90	90	m_can_3_int0	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_3_IR[0] & ~M_CAN_3_ILS[0])| ... | (M_CAN_3_IR[11] & ~M_CAN_3_ILS[11]) | (M_CAN_3_IR[13] & ~M_CAN_3_ILS[13]) | … | (M_CAN_3_IR[31] & ~M_CAN_3_ILS[31])	0x0000 01A8
Settable	91	91	m_can_3_int1	Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_3_IR[0] & M_CAN_3_ILS[0])| ... | (M_CAN_3_IR[11] & M_CAN_3_ILS[11]) | (M_CAN_3_IR[13] & M_CAN_3_ILS[13]) | … | (M_CAN_3_IR[31] & M_CAN_3_ILS[31])	0x0000 01AC
Settable	92	92	mcan_3_ dmu_int	MCAN3 DMU Interrupt	0x0000 01B0

Table 16. SR5E1x vector table (continued)

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
Settable	93	93	m_can_4_int0	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_4_IR[0] & ~M_CAN_4_ILS[0])| ... | (M_CAN_4_IR[11] & ~M_CAN_4_ILS[11]) | (M_CAN_4_IR[13] & ~M_CAN_4_ILS[13]) | … | (M_CAN_4_IR[31] & ~M_CAN_4_ILS[31])	0x0000 01B4
Settable	94	94	m_can_4_int1	Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_4_IR[0] & M_CAN_4_ILS[0])| ... | (M_CAN_4_IR[11] & M_CAN_4_ILS[11]) | (M_CAN_4_IR[13] & M_CAN_4_ILS[13]) | … | (M_CAN_4_IR[31] & M_CAN_4_ILS[31])	0x0000 01B8
Settable	95	95	mcan_4_ dmu_int	MCAN3 DMU Interrupt	0x0000 01BC
Settable	96	96	Cordic	Cordic Interrupt	0x0000 01C0
Settable	97	97	—	Reserved	0x0000 01C4
Settable	98	98	HSEM_int1	Hardware semaphore interrupt to CORE1	0x0000 01C8
Settable	99	99	HSEM_int2	Hardware semaphore interrupt to CORE2	0x0000 01CC
Settable	100	100	—	Reserved	0x0000 01D0
Settable	101	101	—	Reserved	0x0000 01D4
Settable	102	102	—	Reserved	0x0000 01D8
Settable	103	103	—	Reserved	0x0000 01DC
Settable	104	104	HRTIM2_Master_Int	HRTIM2 Master Timer Interrupt (hrtim_it1)	0x0000 01E0
Settable	105	105	HRTIM2_TIMA_Int	HRTIM2 Timer A Interrupt (hrtim_it2)	0x0000 01E4
Settable	106	106	HRTIM2_TIMB_Int	HRTIM2 Timer B Interrupt (hrtim_it3)	0x0000 01E8
Settable	107	107	HRTIM2_TIMC_Int	HRTIM2 Timer C Interrupt (hrtim_it4)	0x0000 01EC
Settable	108	108	HRTIM2_TIMD_Int	HRTIM2 Timer D Interrupt (hrtim_it5)	0x0000 01F0
Settable	109	109	HRTIM2_TIME_Int	HRTIM2 Timer E Interrupt (hrtim_it6)	0x0000 01F4
Settable	110	110	HRTIM2_TIM_FLT_Int	HRTIM2 Fault Interrupt (hrtim_it8)	0x0000 01F8
Settable	111	111	HRTIM2_TIMF_Int	HRTIM2 Timer F Interrupt (hrtim_it7)	0x0000 01FC
Settable	112	112	—	Reserved	0x0000 0200

Table 16. SR5E1x vector table (continued)

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
Settable	113	113	—	Reserved	0x0000 0204
Settable	114	114	—	Reserved	0x0000 0208
Settable	115	115	—	Reserved	0x0000 020C
Settable	116	116	—	Reserved	0x0000 0210
Settable	117	117	—	Reserved	0x0000 0214
Settable	118	118	—	Reserved	0x0000 0218
Settable	119	119	—	Reserved	0x0000 021C
Settable	120	120	B-DAC1	B-DAC1 underrun global interrupt	0x0000 0220
Settable	121	121	DAC1	DAC1 underrun global interrupt	0x0000 0224
Settable	122	122	DAC2	DAC2 underrun global interrupt	0x0000 0228
Settable	123	123	DAC3	DAC3 underrun global interrupt	0x0000 022C
Settable	124	124	DAC4	DAC4 underrun global interrupt	0x0000 0230
Settable	125	125	—	Reserved	0x0000 0234
Settable	126	126	—	Reserved	0x0000 0238
Settable	127	127	—	Reserved	0x0000 023C
Settable	128	128	—	Reserved	0x0000 0240
Settable	129	129	COMP_1_2	COMP1 OR COMP2 through EXTI lines 21/22 interrupts	0x0000 0244
Settable	130	130	COMP_3_4	COMP3 OR COMP4 through EXTI lines 29/30 interrupts	0x0000 0248
Settable	131	131	COMP_5_6	COMP5 OR COMP6 through EXTI lines 31/32 interrupts	0x0000 024C
Settable	132	132	COMP7_8	COMP7 OR COMP8 through EXTI lines 33/34 interrupts	0x0000 0250
Settable	133	133	—	Reserved	0x0000 0254
Settable	134	134	—	Reserved	0x0000 0258
Settable	135	135	—	Reserved	0x0000 025C
Settable	136	136	—	Reserved	0x0000 0260
Settable	137	137	ADC1	ADC1 global interrupt	0x0000 0264
Settable	138	138	ADC2	ADC2 global interrupt	0x0000 0268
Settable	139	139	ADC3	ADC3 global interrupt	0x0000 026C
Settable	140	140	ADC4	ADC4 global interrupt	0x0000 0270

Table 16. SR5E1x vector table (continued)

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
Settable	141	141	ADC5	ADC5 global interrupt	0x0000 0274
Settable	142	142	—	Reserved	0x0000 0278
Settable	143	143	—	Reserved	0x0000 027C
Settable	144	144	—	Reserved	0x0000 0280
Settable	145	145	SD_ADC1	SD_ADC1 global interrupt	0x0000 0284
Settable	146	146	SD_ADC2	SD_ADC2 global interrupt	0x0000 0288
Settable	147	147	—	Reserved	0x0000 028C
Settable	148	148	—	Reserved	0x0000 0290
Settable	149	149	—	Reserved	0x0000 0294
Settable	150	—	Core2_SEV	Cortex M7 Core2 Send Event interrupt to Core1 through EXTI line 45	0x0000 0298
Settable	—	151	Core1_SEV	Cortex M7 Core1 Send Event interrupt to Core2 through EXTI line 44	0x0000 029C
Settable	152	152	—	Reserved	0x0000 02A0
Settable	153	153	—	Reserved	0x0000 02A4
Settable	154	154	FCCU_int	FCCU_IRQ_STAT[ALARM_STAT or CFG_TO_STAT]	0x0000 02A8
Settable	155	155	—	Reserved	0x0000 02AC
Settable	156	156	—	Reserved	0x0000 02B0
Settable	157	157	—	Reserved	0x0000 02B4
Settable	158	158	—	Reserved	0x0000 02B8
Settable	159	159	—	Reserved	0x0000 02BC
Settable	160	160	PMC_dig	PMC_dig global interrupt	0x0000 02C0
Settable	161	161	—	Reserved	0x0000 02C4
Settable	162	162	—	Reserved	0x0000 02C8
Settable	163	163	—	Reserved	0x0000 02CC
Settable	164	164	HSM2HST0_15	Ored(HSM2HT_it[15]...[HSM2HT_it[0])	0x0000 02D0
Settable	165	165	HSM2HST16_31	Ored(HSM2HT_it[31]...[HSM2HT_it[16])	0x0000 02D4
Settable	166	166	—	Reserved	0x0000 02D8

Table 16. SR5E1x vector table (continued)

Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
Settable	167	167	NVMPC	Requested Read from CoreX in Data Flash available (Data memory collision function)	0x0000 02DC
Settable	168	168	DBGMCU	DBGMCU_MSR[DIN_INT] :OR: DBGMCU_MSR[DOUT_INT]	0x0000 02E0
Settable	169	169	—	Reserved	0x0000 02E4
Settable	170	170	—	Reserved	0x0000 02E8
Settable	171	171	—	Reserved	0x0000 02EC
Settable	172	172	—	Reserved	0x0000 02F0
Settable	173	173	—	Reserved	0x0000 02F4
Settable	174	174	—	Reserved	0x0000 02F8
Settable	175	175	—	Reserved	0x0000 02FC
Settable	176	176	—	Reserved	0x0000 0300
Settable	177	177	—	Reserved	0x0000 0304
Settable	178	178	—	Reserved	0x0000 0308
Settable	179	179	—	Reserved	0x0000 030C
Settable	180	180	—	Reserved	0x0000 0310
Settable	181	181	—	Reserved	0x0000 0314
Settable	182	182	—	Reserved	0x0000 0318
Settable	183	183	—	Reserved	0x0000 031C
Settable	184	184	—	Reserved	0x0000 0320
Settable	185	185	—	Reserved	0x0000 0324
Settable	186	186	—	Reserved	0x0000 0328
Settable	187	187	—	Reserved	0x0000 032C
Settable	188	188	—	Reserved	0x0000 0330
Settable	189	189	—	Reserved	0x0000 0334
Settable	190	190	—	Reserved	0x0000 0338
Settable	191	191	—	Reserved	0x0000 033C
Settable	192	192	—	Reserved	0x0000 0340
Settable	193	193	—	Reserved	0x0000 0344
Settable	194	194	—	Reserved	0x0000 0348
Settable	195	195	—	Reserved	0x0000 034C
Settable	196	196	—	Reserved	0x0000 0350

Table 16. SR5E1x vector table (continued)

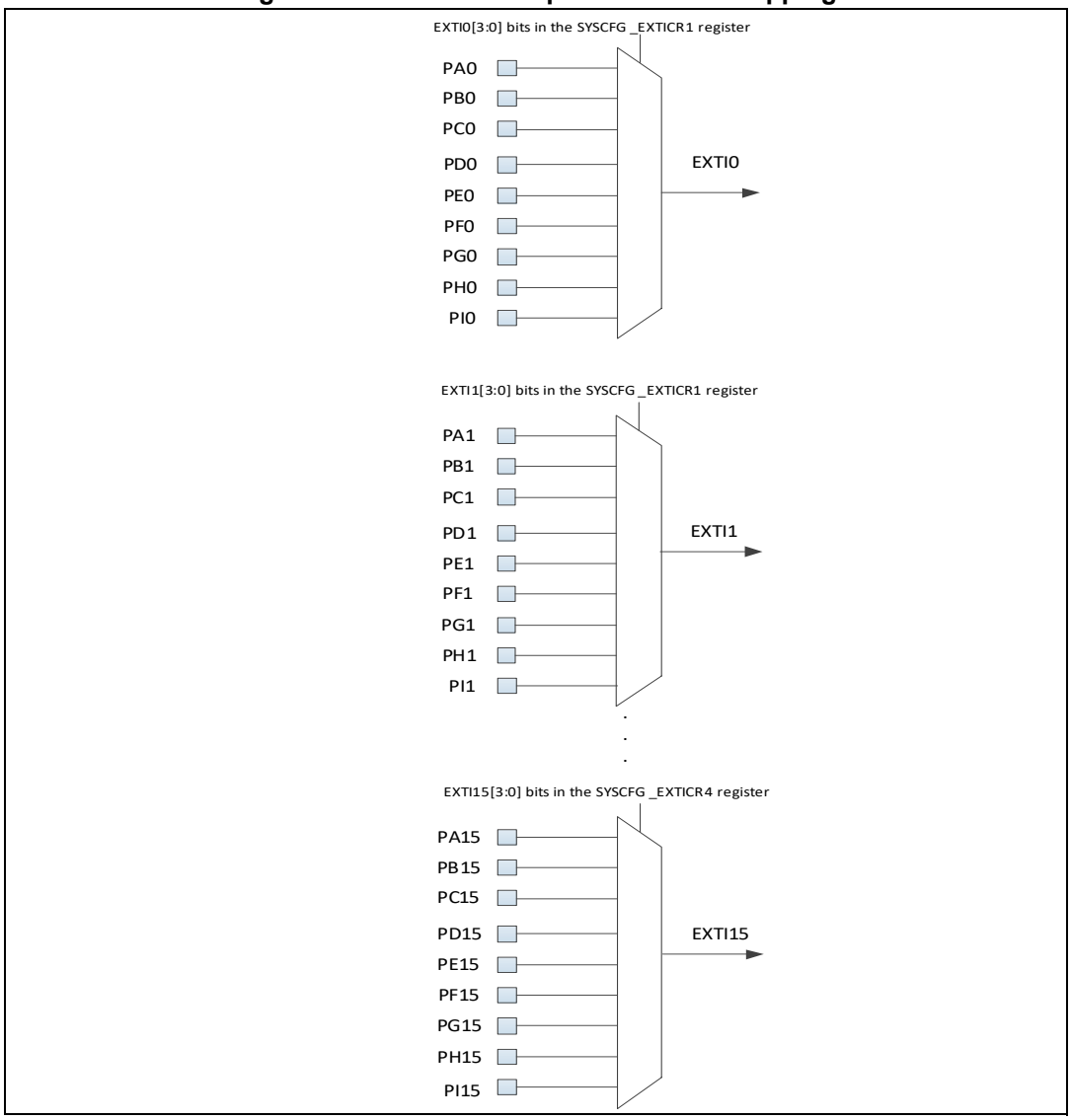
Type of priority	NVIC1 Position	NVIC2 Position	Acronym	Description	Address
Settable	197	197	—	Reserved	0x0000 0354
Settable	198	198	—	Reserved	0x0000 0358
Settable	199	199	—	Reserved	0x0000 035C
Settable	200	200	—	Reserved	0x0000 0360

5.2.5 EXTI configuration

The SR5E1x EXTI features 33 interrupt/event lines.

In the following figure, the GPIOs are connected to 16 configurable interrupt/event lines.

Figure 7. External interrupt/event GPIO mapping



The EXTI lines are connected as shown in the following table.

Table 17. EXTI event input mapping

Event input	Source	Event input type	Connection to NVICx
0-15	GPIOs	Configurable	1,2
16-17	Reserved	—	—
18	RTC alarm event	Configurable	1,2
19	Reserved	—	—
20	RTC Wakeup Timer	Configurable	1,2
21	COMP1 Output	Configurable	1,2
22	COMP2 Output	Configurable	1,2

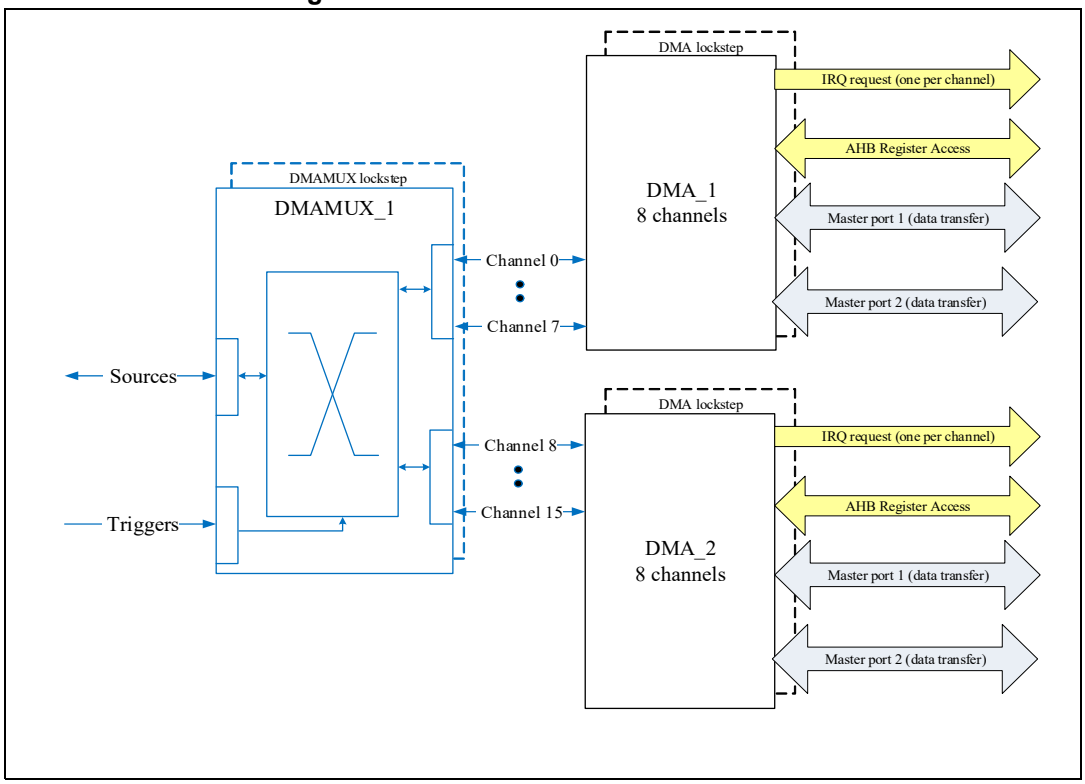
Table 17. EXTI event input mapping (continued)

Event input	Source	Event input type	Connection to NVICx
23	I2C1 wakeup	Direct	1,2
24	I2C2 wakeup	Direct	1,2
25	UART1	Direct	1,2
26	UART2	Direct	1,2
27	Reserved	—	—
28	UART3	Direct	1,2
29	COMP3 Output	Configurable	1,2
30	COMP4 Output	Configurable	1,2
31	COMP5 Output	Configurable	1,2
32	COMP6 Output	Configurable	1,2
33	COMP7 Output	Configurable	1,2
34	COMP8 Output	Configurable	1,2
35-43	Reserved	—	—
44	CPU1 SEV	Direct	2
45	CPU2 SEV	Direct	1
46	Reserved	—	—

5.2.6 DMA controller configuration

The SR5E1x includes 2 Direct Memory Access (DMA) instances (DMA1 and DMA2) and a DMA request multiplexer (DMAMUX). The *Figure 8* below shows the DMA architecture.

Figure 8. DMA and DMAMUX architecture



Each DMA instance has the following features:

- 8 input streams
- double buffer mode for each stream
- 4-word FIFO per stream

Each DMA instance has a replica that can be enabled in lockstep in order to reach a high level of safety. The enabling of the replica is done at start-up via the Global System Configuration DCF record.

5.2.7 DMA request multiplexer (DMAMUX) configuration

The SR5E1x includes one DMA request multiplexer (DMAMUX_1) with 16 output channels:

- DMAMUX_1 channels 0 to 7 are connected to DMA_1 channels 0 to 7
- DMAMUX_1 channels 8 to 15 are connected to DMA_2 channels 0 to 7

The DMAMUX_1 has a replica that can be enabled in lockstep at start-up via the Global System Configuration DCF record.

5.2.7.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

Table 18. DMAMUX instantiation

Feature	DMAMUX
Number of DMAMUX output request channels	16
Number of DMAMUX request generator channels	4
Number of DMAMUX request trigger inputs	21
Number of DMAMUX synchronization inputs	21
Number of DMAMUX peripheral request inputs	151(1)

1. Inputs 1 to 151 used but 33 unused inputs inside.

5.2.7.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

DMAMUX is used with DMA1 and DMA2:

- DMAMUX channels 0 to 7 are connected to DMA1 input streams 0 to 7.
- DMAMUX channels 8 to 15 are connected to DMA2 input streams 0 to 7.

Table 19. DMAMUX: assignment of multiplexer inputs to resources

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	DMAMUX_Req G0	55	TIM8_COM	109	Reserved
2	DMAMUX_Req G1	56	TIM2_CH1	110	Reserved
3	DMAMUX_Req G2	57	TIM2_CH2	111	Reserved
4	DMAMUX_Req G3	58	TIM2_CH3	112	Cordic_read
5	ADC1	59	TIM2_CH4	113	Cordic_write
6	B-DAC1_CH1	60	TIM2_UP	114	Reserved
7	B-DAC1_CH2	61	TIM3_CH1	115	Reserved
8	TIM6_UP	62	TIM3_CH2	116	HRTIM2_MASTER (hrtim_dma1)
9	TIM7_UP	63	TIM3_CH3	117	HRTIM2_TIMA (hrtim_dma2)
10	SPI1_RX	64	TIM3_CH4	118	HRTIM2_TIMB (hrtim_dma3)
11	SPI1_TX	65	TIM3_UP	119	HRTIM2_TIMC (hrtim_dma4)
12	SPI2_RX	66	TIM3_TRIG	120	HRTIM2_TIMD (hrtim_dma5)
13	SPI2_TX	67	TIM4_CH1	121	HRTIM2_TIME (hrtim_dma6)
14	SPI3_RX	68	TIM4_CH2	122	HRTIM2_TIMF (hrtim_dma7)
15	SPI3_TX	69	TIM4_CH3	123	Reserved
16	I2C1_RX	70	TIM4_CH4	124	Reserved
17	I2C1_TX	71	TIM4_UP	125	Reserved
18	I2C2_RX	72	TIM5_CH1	126	Reserved
19	I2C2_TX	73	TIM5_CH2	127	Reserved

Table 19. DMAMUX: assignment of multiplexer inputs to resources (continued)

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
20	Reserved	74	TIM5_CH3	128	Reserved
21	Reserved	75	TIM5_CH4	129	Reserved
22	Reserved	76	TIM5_UP	130	CAN_SUB_1_DMU_1_TX
23	Reserved	77	TIM5_TRIG	131	CAN_SUB_1_DMU_1_TXE
24	UART1_RX	78	TIM15_CH1	132	CAN_SUB_1_DMU_1_RX0
25	UART1_TX	79	TIM15_UP	133	CAN_SUB_1_DMU_1_RX1
26	UART2_RX	80	TIM15_TRIG	134	CAN_SUB_1_DMU_2_TX
27	UART2_TX	81	TIM15_COM	135	CAN_SUB_1_DMU_2_TXE
28	UART3_RX	82	TIM16_CH1	136	CAN_SUB_1_DMU_2_RX0
29	UART3_TX	83	TIM16_UP	137	CAN_SUB_1_DMU_2_RX1
30	Reserved	84	TIM_TS_UP	138	CAN_SUB_1_DMU_3_TX
31	Reserved	85	Reserved	139	CAN_SUB_1_DMU_3_TXE
32	Reserved	86	Reserved	140	CAN_SUB_1_DMU_3_RX0
33	Reserved	87	Reserved	141	CAN_SUB_1_DMU_3_RX1
34	Reserved	88	Reserved	142	CAN_SUB_1_DMU_4_TX
35	Reserved	89	Reserved	143	CAN_SUB_1_DMU_4_TXE
36	ADC2	90	Reserved	144	CAN_SUB_1_DMU_4_RX0
37	ADC3	91	DAC1_CH1	145	CAN_SUB_1_DMU_4_RX1
38	ADC4	92	DAC1_CH2	146	CAN_SUB_1_M_CAN_1
39	ADC5	93	DAC2_CH1	147	CAN_SUB_1_M_CAN_2
40	Reserved	94	DAC2_CH2	148	Reserved
41	Reserved	95	HRTIM1_MASTER (hrtim_dma1)	149	Reserved
42	TIM1_CH1	96	HRTIM1_TIMA (hrtim_dma2)	150	SD_ADC1
43	TIM1_CH2	97	HRTIM1_TIMB (hrtim_dma3)	151	SD_ADC2
44	TIM1_CH3	98	HRTIM1_TIMC (hrtim_dma4)	152	Reserved
45	TIM1_CH4	99	HRTIM1_TIMD (hrtim_dma5)	153	Reserved
46	TIM1_UP	100	HRTIM1_TIME (hrtim_dma6)	154	Reserved
47	TIM1_TRIG	101	HRTIM1_TIMF (hrtim_dma7)	155	Reserved
48	TIM1_COM	102	DAC3_CH1	156	Reserved

Table 19. DMAMUX: assignment of multiplexer inputs to resources (continued)

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
49	TIM8_CH1	103	DAC3_CH2	157	Reserved
50	TIM8_CH2	104	DAC4_CH1	158	Reserved
51	TIM8_CH3	105	DAC4_CH2	159	Reserved
52	TIM8_CH4	106	SPI4_RX	160	Reserved
53	TIM8_UP	107	SPI4_TX	-	—
54	TIM8_TRIG	108	Reserved	-	—

Table 20. DMAMUX: assignment of trigger inputs to resources

Trigger input	Resource	Trigger input	Resource
0	EXTI LINE0	16	DMAMUX1_ch0_event
1	EXTI LINE1	17	DMAMUX1_ch1_event
2	EXTI LINE2	18	DMAMUX1_ch2_event
3	EXTI LINE3	19	DMAMUX1_ch3_event
4	EXTI LINE4	20	Reserved
5	EXTI LINE5	21	Reserved
6	EXTI LINE6	22	Reserved
7	EXTI LINE7	23	Reserved
8	EXTI LINE8	24	Reserved
9	EXTI LINE9	25	Reserved
10	EXTI LINE10	26	Reserved
11	EXTI LINE11	27	Reserved
12	EXTI LINE12	28	Reserved
13	EXTI LINE13	29	Reserved
14	EXTI LINE14	30	Reserved
15	EXTI LINE15	31	Reserved

Table 21. DMAMUX: assignment of synchronization inputs to resources

Sync. input	Resource	Sync. input	Resource
0	EXTI LINE0	16	DMAMUX1_ch0_event
1	EXTI LINE1	17	DMAMUX1_ch1_event
2	EXTI LINE2	18	DMAMUX1_ch2_event
3	EXTI LINE3	19	DMAMUX1_ch3_event
4	EXTI LINE4	20	Reserved
5	EXTI LINE5	21	Reserved
6	EXTI LINE6	22	Reserved

Table 21. DMAMUX: assignment of synchronization inputs to resources (continued)

Sync. input	Resource	Sync. input	Resource
7	EXTI LINE7	23	Reserved
8	EXTI LINE8	24	Reserved
9	EXTI LINE9	25	Reserved
10	EXTI LINE10	26	Reserved
11	EXTI LINE11	27	Reserved
12	EXTI LINE12	28	Reserved
13	EXTI LINE13	29	Reserved
14	EXTI LINE14	30	Reserved
15	EXTI LINE15	31	Reserved

5.2.8 GPIO configuration

The General-Purpose I/O modules control the MCU pad configuration, ports, general-purpose input and output (GPIO) signals and alternate function configuration.

The I/Os are organized as 16-bit ports: ports A to I, also known as GPIOA to GPIOI. GPIOG contains 13 pins from channel 3 to 15. GPIOI contains 9 pins from channel 0 to 9. Up to 135 user pins are available in QFP176, 103 user pins in QFP144 and 61 user pins in QFP100.

For more details, refer to *Chapter 20: General-purpose I/Os (GPIO)*.

5.3 Memories and memory interfaces

See the following chapters for detailed information on memories and memory interfaces.

Table 22. Memories and memory interface references

Topic	Reference
System memory map	*Chapter 3: System and memory overview*
Flash controller	*Chapter 16: Nonvolatile memory platform controller (NVMPC)*
Embedded flash memory	*Chapter 17: Embedded flash memory*
Platform RAM controller	*Chapter 15: Platform RAM controller AXI (PRAMC_AXI)*

5.3.1 Non-volatile memory platform controller (NVMPC) configuration

Two non-volatile memory controllers and arrays are implemented. The NVMPCs provide flash configuration and control functions and manage the interface between the flash memory array and the bus matrix.

5.3.2 Platform RAM controller (PRAMC)

Two RAM controllers and arrays are implemented. The PRAMCs provide RAM configuration and manage the interface between the SRAM arrays and the system bus.

5.4 Analog modules

5.4.1 Temperature sensor configuration

The temperature sensor is used to measure the temperature of the SR5E1x device. The temperature sensor has the following outputs:

- Two analog voltages
  - A voltage signal that is linearly increasing with internal junction temperature (PTAT that is proportional to absolute temperature)
  - A voltage signal that is linearly decreasing with internal junction temperature (CTAT that is complementary proportional to absolute temperature)
- Three digital outputs that signal under- or overtemperature operating conditions.

During production testing of the device, the output voltages of the temperature sensor are sampled by SARADCx of the onboard ADC module, (refer to *Table 23: ADCx connectivity (x = 1..5)*, at a predefined high temperature and also at a predefined low temperature. These calibration parameters are stored during production test into UTEST Flash memory (See Section UTEST in Chapter Flash Memory.)

By converting the two analog outputs of the temperature sensor through ADC, one can find out a constant reference code which does not change with temperature or the ADC reference values. It is also called 'Digital Bandgap Voltage'. See *Section 33.3.2: Equations for converting TSENS voltages into constant reference (Digital bandgap voltage)* in *Chapter 33: Temperature sensor*.

For more details, see *Chapter 33: Temperature sensor*.

5.4.2 ADCx configuration

ADCx inputs are connected to the external channels as well as internal sources as described in the following table.

Table 23. ADCx connectivity (x = 1..5)

ADCx channel number	ADCx channel type	ADCx Analog inputs	ADC channel source assignment				
			ADC1	ADC2	ADC3	ADC4	ADC5
ADCx_IN0	fast	Vinp[0]	Vssa	Vssa	Vssa	Vssa	Vssa
		Vinn[0]	Vssa	Vssa	Vssa	Vssa	Vssa

Table 23. ADCx connectivity (x = 1..5) (continued)

ADCx channel number	ADCx channel type	ADCx Analog inputs	ADC channel source assignment				
			ADC1	ADC2	ADC3	ADC4	ADC5
ADCx_IN1	fast	Vinp[1]	PB2	PB7	PB12	PC3	PC9
		Vinn[1]	PB3	PB8	PB13	PC4	PC10
ADCx_IN2	fast	Vinp[2]					
		Vinn[2]	PB4	PB9	PB14	PC6	PC11
ADCx_IN3	fast	Vinp[3]					
		Vinn[3]	PB5	PB10	PB15	PC7	PC12
ADCx_IN4	fast	Vinp[4]					
		Vinn[4]	PB6	PB11	PC0	PC8	PC13
ADCx_IN5	fast	Vinp[5]					
		Vinn[5]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN6	slow	Vinp[6]	PD3	PD6	PC14	PD1	PE6
		Vinn[6]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN7	slow	Vinp[7]	PD4	PD9	PC15	PD2	PE7
		Vinn[7]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN8	slow	Vinp[8]	PD5	PE0	PD0	PE5	PE8
		Vinn[8]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN9	slow	Vinp[9]	PC1 (SAR_CAL1) (1)	PC1 (SAR_CAL1) (1)	PC1 (SAR_CAL1) (1)	PC1 (SAR_CAL1) (1)	PC1 (SAR_CAL1) (1)
		Vinn[9]	PC2 (SAR_CAL2) (2)	PC2 (SAR_CAL2) (2)	PC2 (SAR_CAL2) (2)	PC2 (SAR_CAL2) (2)	PC2 (SAR_CAL2) (2)
ADCx_IN10	slow	Vinp[10]					
		Vinn[10]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN11	slow	Vinp[11]	PC7	PB15	PB10	PB5	Vdd_lv
		Vinn[11]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN12	slow	Vinp[12]	PC8	PC0	PB11	PB6	Vdd_hv
		Vinn[12]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN13	slow	Vinp[13]	TempSens CTAT	Vdd_lv	B-DAC_CH1	Vdd_lv	TempSens CTAT
		Vinn[13]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN14	slow	Vinp[14]	TempSens PTAT	Vdd_hv	B-DAC_CH2	Vdd_hv	TempSens PTAT
		Vinn[14]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN15	slow	Vinp[15]	PC6	PB12	PB9	PB4	B-DAC_CH1
		Vinn[15]	Vssa	Vssa	Vssa	Vssa	Vssa

Table 23. ADCx connectivity (x = 1..5) (continued)

ADCx channel number	ADCx channel type	ADCx Analog inputs	ADC channel source assignment				
			ADC1	ADC2	ADC3	ADC4	ADC5
ADCx_IN16	slow	Vinp[16]	PA13	PB13	PA15	PB2	PB7
		Vinn[16]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN17	slow	Vinp[17]	PA14	PB14	PB0	PB3	PB8
		Vinn[17]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN18	Bias test(3)	Vinp[18]	Vref, Vref/3, Vref*2/3	Vref, Vref/3, Vref*2/3	Vref, Vref/3, Vref*2/3	Vref, Vref/3, Vref*2/3	Vref, Vref/3, Vref*2/3
		Vinn[18]	Vssa	Vssa	Vssa	Vssa	Vssa
ADCx_IN19	slow	Vinp[19]	Reserved	Reserved	Reserved	Reserved	Reserved
		Vinn[19]	Reserved	Reserved	Reserved	Reserved	Reserved

1. SAR_CAL1 means Common Analog Level 1 for all SAR instances.
2. SAR_CAL2 means Common Analog Level 2 for all SAR instances.
3. BIAS test uses the “internal reference voltages” selected by the SYS_CFG.ST_VREF_SEL2 register. For details, refer to the System configuration controller (SYSCFG) chapter.

Table 24. ADCs trigger selection

ADC trigger selection EXTSEL[4:0] or JEXTSEL[4:0]	ADC trigger signal assignment			
	ADC 1/2		ADC3/4/5	
	Regular	Injected	Regular	Injected
0	tim1_cc1	tim1_trgo	tim3_cc1	tim1_trgo
1	tim1_cc2	tim1_cc4	tim3_cc1	tim1_cc4
2	tim1_cc3	tim2_trgo	tim2_cc3	tim2_trgo
3	tim2_cc2	tim2_cc1	tim8_cc1	tim8_cc2
4	tim3_trgo	tim3_cc4	tim3_trgo	tim4_cc3
5	tim4_cc4	tim4_trgo	exti2	tim4_trgo
6	exti11	exti15	tim4_cc1	tim4_cc4
7	tim8_trgo	tim8_cc4	tim8_trgo	tim8_cc4
8	tim8_trgo2	tim1_trgo2	tim8_trgo2	tim1_trgo2
9	tim1_trgo	tim8_trgo	tim1_trgo	tim8_trgo
10	tim1_trgo2	tim8_trgo2	tim1_trgo2	tim8_trgo2
11	tim2_trgo	tim3_cc3	tim2_trgo	tim1_cc3
12	tim4_trgo	tim3_trgo	tim4_trgo	tim3_trgo
13	tim6_trgo	tim3_cc1	tim6_trgo	exti3
14	tim15_trgo	tim6_trgo	tim15_trgo	tim6_trgo
15	tim3_cc4	tim15_trgo	tim2_cc1	tim15_trgo

Table 24. ADCs trigger selection (continued)

ADC trigger selection EXTSEL[4:0] or JEXTSEL[4:0]	ADC trigger signal assignment			
	ADC 1/2		ADC3/4/5	
	Regular	Injected	Regular	Injected
16	hrtim1_adc_trg1	hrtim1_adc_trg2	hrtim1_adc_trg1	hrtim1_adc_trg2
17	hrtim1_adc_trg3	hrtim1_adc_trg4	hrtim1_adc_trg3	hrtim1_adc_trg4
18	hrtim1_adc_trg5	hrtim1_adc_trg5	hrtim1_adc_trg5	hrtim1_adc_trg5
19	hrtim1_adc_trg6	hrtim1_adc_trg6	hrtim1_adc_trg6	hrtim1_adc_trg6
20	hrtim1_adc_trg7	hrtim1_adc_trg7	hrtim1_adc_trg7	hrtim1_adc_trg7
21	hrtim1_adc_trg8	hrtim1_adc_trg8	hrtim1_adc_trg8	hrtim1_adc_trg8
22	hrtim1_adc_trg9	hrtim1_adc_trg9	hrtim1_adc_trg9	hrtim1_adc_trg9
23	hrtim1_adc_trg10	hrtim1_adc_trg10	hrtim1_adc_trg10	hrtim1_adc_trg10
24	hrtim2_adc_trg1	hrtim2_adc_trg2	hrtim2_adc_trg1	hrtim2_adc_trg2
25	hrtim2_adc_trg3	hrtim2_adc_trg4	hrtim2_adc_trg3	hrtim2_adc_trg4
26	hrtim2_adc_trg5	hrtim2_adc_trg5	hrtim2_adc_trg5	hrtim2_adc_trg5
27	hrtim2_adc_trg6	hrtim2_adc_trg6	hrtim2_adc_trg6	hrtim2_adc_trg6
28	hrtim2_adc_trg7	hrtim2_adc_trg7	hrtim2_adc_trg7	hrtim2_adc_trg7
29	hrtim2_adc_trg8	hrtim2_adc_trg8	hrtim2_adc_trg8	hrtim2_adc_trg8
30	hrtim2_adc_trg9	hrtim2_adc_trg9	hrtim2_adc_trg9	hrtim2_adc_trg9
31	hrtim2_adc_trg10	hrtim2_adc_trg10	hrtim2_adc_trg10	hrtim2_adc_trg10

5.4.3 SDADC configuration

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides the accurate conversion data for a wide range of applications.

SR5E1x includes two independent 16-bit SDADCs.

Each module features two inputs configurable as two single-ended channels or one differential channel.

5.4.3.1 Sigma-delta ADC clock sources

Each Sigma-Delta ADC digital interface has a register clock input, which is separate from the module clock input. The register interface is clocked by the APB2 bridge clock PCLK2. The module clock is sourced by the modulator clock (SD_CLK). The input clock for the Sigma-Delta modulator is the SD_CLK.

Both the analog and digital blocks of the ADC contain an internal clock Mux for external modulator mode. The external modulator clock (EXT_CLK) can be input to the device, or generated from SD_CLK from the RCC. The pad input/output settings in the corresponding GPIOx pad control registers determine the external modulator clock source to the ADC blocks. The EXT_CLK pins are multiplexed on multiple external pins on the device.

Selection of the EXT_CLK pin to use for the external modulator clock for each ADC is done in the GPIOx configuration registers. Each Sigma-Delta ADC has two external pins to select from for an external modulator clock.

5.4.3.2 SDADC analog input channel selection

For the Sigma-Delta Mux inputs AN[x], the positive and negative input terminals are selected in the Sigma-Delta ADC digital interface Channel Selection Register (CSR).

This section shows analog channel selections for each combination of module configuration.

Register (MCR) field values MCR_MODE and MCR_VCOMSEL. The following table provides a quick reference key to the MCR fields used in the SD ADC analog input selection tables that follow it.

Table 25. Register fields for SDADC analog input selection

Field	Description
MCR_VCOMSEL	Common voltage bias selection This bit selects the common voltage bias for the negative input terminal of SD ADC during single-ended mode (MODE=1). 0 Negative input terminal is biased with VREFN. 1 Negative input terminal is biased with VREFP/2 (half scale bias).
MCR_MODE	Mode selection 0 Differential input mode is selected. 1 Single-ended input mode is selected.
CSR_ANCHSEL	Analog channel selection Based on single-ended or differential mode of operation (MODE bit of MCR), common mode voltage selection (VCOMSEL bit of MCR), this bit field defines the connectivity of analog inputs to either positive or negative polarity terminals of Sigma-Delta ADC, as shown in the following tables: – *Table 26* for channel selection – *Table 27* for input source selection

Table 26. SDADC analog input selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		111 .. 010	reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		111 .. 010	reserved	

Table 26. SDADC analog input selection (continued)

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (negative terminal)
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 27. SDADCx analog input source

SDADC channel number	SDADC1 pin name	SDADC2 pin name
AN[0]	SDADC1_IN0 (PA13)	SDADC2_IN0 (PA15)
AN[1]	SDADC1_IN1 (PA14)	SDADC2_IN1 (PB0)

5.4.3.3 SDADC starting/triggering

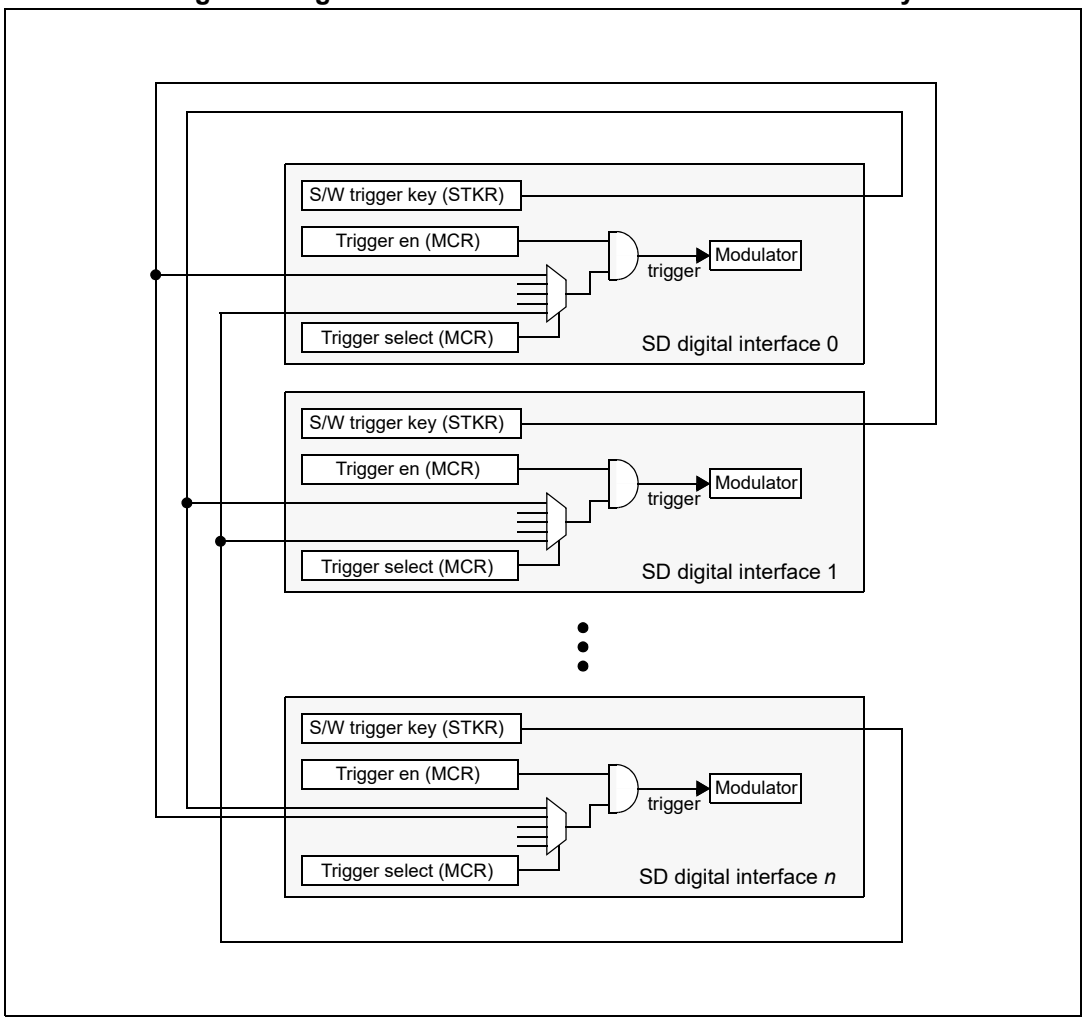
Each Sigma-Delta ADC supports starting the converter with a software write to an enable bit in a register in the digital interface.

A method to start multiple Sigma-Delta converters simultaneously by a single software write to a control register is available on the device. In this case, “simultaneously” means that all selected converters start sampling their inputs on the same SD_CLK clock edge.

The digital interface of the Sigma-Delta ADC contains the control register for simultaneous start of the ADCs. When the register is written to start multiple converters, an output from the digital interface is asserted. The hardware trigger input of each Sigma-Delta ADC is connected to this simultaneous start output. This way, the user controls which ADCs are selected by enabling the hardware trigger of the desired ADCs before writing to the starting simultaneous converters.

The connections for the simultaneous start of the converters are given in the figure below.

Figure 9. Sigma-Delta ADC simultaneous start connectivity



5.4.3.4 Sigma-delta mux input advance and wraparound

The Mux wraparound feature is enabled and controlled in the digital interface of the Sigma-Delta ADC. The advancing of the Mux can be controlled in the interface by software only.

5.4.3.5 Conversion result interrupt/DMA

A DMA or interrupt can be generated for each conversion for each ADC. The Sigma-Delta ADC has an 8-deep result FIFO.

The Sigma-Delta ADC digital interface has an input signal that is used to disable interrupt and DMA requests for conversion results. The input signal is connected to outputs from the timers modules. The digital interface has a register bit for enabling and disabling the interrupt/DMA gating feature.

When the signal from a timer is asserted, no interrupt or DMA request is generated for any conversion results. The polarity of the gating signal is such that when the signal is high, interrupt and DMA requests are passed. When the gating signal is low, interrupt and DMA requests are blocked.

The gating signal for each ADC is driven by several signals coming from the timer modules as listed in *Table 28*. The selection is done via the register SDADCx_EXT_GATE_SEL of the System Configuration controller (SYSCFG).

For both SDADC instances, the interrupt/DMA selectable gating signals are:

- tim1_oc1
- tim1_oc5
- tim8_oc1
- tim8_oc5
- tim2_oc4
- tim3_oc4
- tim4_oc4
- tim5_oc4
- tim15_oc2
- hrtim1_CHE1
- hrtim1_CHF1
- hrtim2_CHE1
- hrtim2_CHF1

5.4.3.6 Conversion limit watchdog output to timers

Each Sigma-Delta ADC provides an optional conversion limit check for upper and lower bounds. The enable bit and the two 16-bit limit registers for the feature are included in the SD ADC wrapper. If a conversion result exceeds one of the limits, a flag bit is asserted in a status register in the SD ADC wrapper. The flag bit is output to TIMx timer input signals for each SD ADC. Refer to the TIMx configuration sections for the interconnection details.

5.4.3.7 SDADC_MCR[TRIGSEL] definitions

The following input triggers apply.

Table 28. SDADC input triggers

SD_ADC_x MCR[TRIGSEL]	Trigger input selection(1)
0000	SD_ADC_1_sw_trig_out: SDADC1 software trigger
0001	SD_ADC_2_sw_trig_out: SDADC2 software trigger
0010	tim1_trgo
0011	tim1_trgo2
0100	tim8_trgo
0101	tim8_trgo2
0110	tim2_trgo
0111	tim3_trgo
1000	tim4_trgo
1001	tim15_trgo
1010	hrtim1_adctrig1

Table 28. SDADC input triggers (continued)

SD_ADC_x MCR[TRIGSEL]	Trigger input selection(1)
1011	hrtim1_adctrig2
1100	hrtim1_adctrig3
1101	hrtim2_adctrig1
1110	hrtim2_adctrig2
1111	hrtim2_adctrig3

1. This field selects which input is used for hardware-triggered conversions

Note: Selecting the input trigger source to be from the output trigger of the same SD ADC is not prohibited by hardware, but is not supported, and results in the SD ADC never responding to the input trigger.

5.4.4 DAC configuration

The SR5E1x embeds five DAC modules. Each module contains two channels.

One module is called B-DAC and provides output buffers allowing to connect the signals outside the device. The other four modules (DAC1... DAC4) signals are connected inside the device.

The DAC digital interface is clocked on the AHB clock.

The DAC digital interface receives the LSIRCOSC/32 clock for sample and hold modes.

5.4.5 Comparator configuration

The device embeds 8 analog comparators. They are grouped by 4 at the register interface level. The implementation is as follows:

- Group1, on AHB2, contains comparators 1 to 4 (named as COMP1 to COMP4 in this document)
- Group2, on AHB1, contains comparators 5 to 8 (named as COMP5 to COMP8)

5.5 Timers

5.5.1 HRTIMx configuration

The SR5E1x includes two instances of High-resolution timer (HRTIM).

Both instances are clocked at the system clock frequency.

5.5.1.1 External events mapping and associated features

There are 2 sets of sources, refer to *Table 29: HRTIM1 external events mapping and associated features* for HRTIM1 and *Table 30: HRTIM2 external events mapping and associated features* for HRTIM2.

Table 29. HRTIM1 external events mapping and associated features

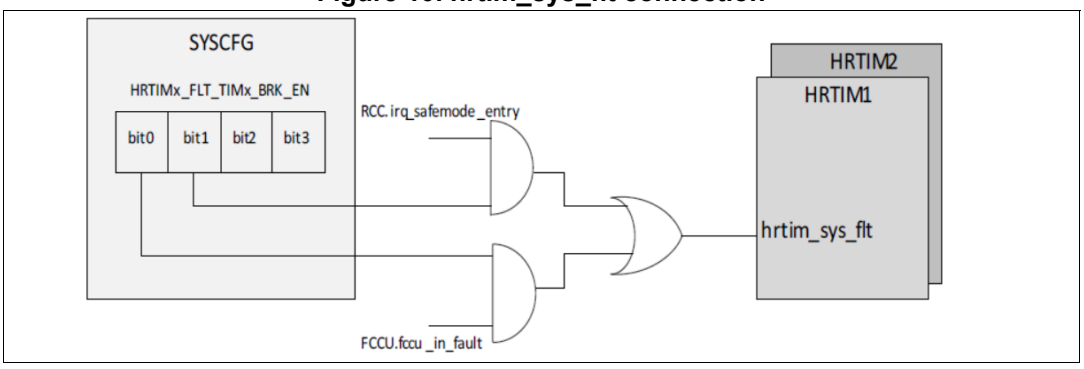
External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E,F	Source (EExSRC[1:0])				Comparator and input sources available per package		
					Src1 (00)	Src 2 (01)	Src3 (10)	Src4 (11)	100-pin	144-pin	176-pin
hrtim_eev1 [4:1]	Yes	—	—	—	HRTIM1_EEV1 (PD4, PG6)	COMP2	TIM1_ TRGO	ADC1_ AWD1	Comp & input (PG6)	Comp & input	Comp & input
hrtim_eev2 [4:1]	Yes	—	—	—	HRTIM1_EEV2 (PD9)	COMP4	TIM2_ TRGO	ADC1_ AWD2	Comp & input	Comp & input	Comp & input
hrtim_eev3 [4:1]	Yes	—	—	—	HRTIM1_EEV3 (PF7, PG3)	COMP6	TIM3_ TRGO	ADC1_ AWD3	Comp & input (PG3)	Comp & input (PG3)	Comp & input
hrtim_eev4 [4:1]	Yes	—	—	—	HRTIM1_EEV4 (PG5)	COMP1	COMP5	ADC2_ AWD1	Comp & input	Comp & input	Comp & input
hrtim_eev5 [4:1]	Yes	—	—	—	HRTIM1_EEV5 (PH7)	COMP3	COMP7	ADC2_ AWD2	Comp	Comp	Comp & input
hrtim_eev6 [4:1]	—	Yes	Yes	—	HRTIM1_EEV6 (PD6, PE0)	COMP2	COMP1	ADC2_ AWD3	Comp	Comp	Comp & input
hrtim_eev7 [4:1]	—	Yes	Yes	—	HRTIM1_EEV7 (PG8)	COMP4	TIM7_ TRGO	ADC3_ AWD1	Comp & input	Comp & input	Comp & input
hrtim_eev8 [4:1]	—	Yes	—	Yes	HRTIM1_EEV8 (PF8, PG4)	COMP6	COMP3	ADC4_ AWD1	Comp & input (PG4)	Comp & input	Comp & input
hrtim_eev9 [4:1]	—	Yes	—	Yes	HRTIM1_EEV9 (PG7)	COMP5	TIM15_ TRGO	COMP4	Comp & input	Comp & input	Comp & input
hrtim_eev10 [4:1]	—	Yes	—	—	HRTIM1_EEV10 (PH7)	COMP7	TIM6_ TRGO	ADC5_ AWD1	—	Comp	Comp & input

Table 30. HRTIM2 external events mapping and associated features

External event channel	Fast mode	Digital filter	Balanc-ed fault timer A,B,C	Balanc-ed fault timer D,E,F	Source (EExSRC[1:0])				Comparator and input sources available per package		
					Src1 (00)	Src 2 (01)	Src3 (10)	Src4 (11)	100-pin	144-pin	176-pin
hrtim_eev1 [4:1]	Yes	—	—	—	HRTIM2_EEV1 (PC15, PG10)	COMP1	TIM1_ TRGO	ADC1_ AWD1	Comp & input (PC15)	Comp & input	Comp & input
hrtim_eev2 [4:1]	Yes	—	—	—	HRTIM2_EEV2 (PF6)	COMP3	TIM2_ TRGO	ADC1_ AWD2	Comp	Comp	Comp & input
hrtim_eev3 [4:1]	Yes	—	—	—	HRTIM2_EEV3 (PF9, PH11)	COMP5	TIM3_ TRGO	ADC1_ AWD3	Comp	Comp & input	Comp & input
hrtim_eev4 [4:1]	Yes	—	—	—	HRTIM2_EEV4 (PG9)	COMP2	COMP6 _OUT	ADC2_ AWD1	Comp	Comp & input	Comp & input
hrtim_eev5 [4:1]	Yes	—	—	—	HRTIM2_EEV5 (PH9)	COMP4	COMP8 _OUT	ADC2_ AWD2	Comp	Comp	Comp & input
hrtim_eev6 [4:1]	—	Yes	Yes	—	HRTIM2_EEV6 (PE13)	COMP1	COMP2 _OUT	ADC2_ AWD3	Comp	Comp	Comp & input
hrtim_eev7 [4:1]	—	Yes	Yes	—	HRTIM2_EEV7 (PG11)	COMP3	TIM7_ TRGO	ADC3_ AWD1	Comp & input	Comp & input	Comp & input
hrtim_eev8 [4:1]	—	Yes	—	Yes	HRTIM2_EEV8 (PF10, PG12)	COMP5	COMP4 _OUT	ADC4_ AWD1	Comp & input (PG12)	Comp & input (PG12)	Comp & input
hrtim_eev9 [4:1]	—	Yes	—	Yes	HRTIM2_EEV9 (PA2)	COMP6	TIM15_ TRGO	COMP4 _OUT	Comp	Comp	Comp & input
hrtim_eev10[4:1]	—	Yes	—	—	HRTIM2_EEV10 (PH9)	COMP8	TIM6_ TRGO	ADC5_ AWD1	—	Comp	Comp & input

5.5.1.2 HRTIM fault inputs

Figure 10. hrtim_sys_flt connection



Each HRTIM has a system fault input (hrtim_sys_flt) connected to the FCCU module and the RCC module.

The hrtim_sys_flt is asserted when the FCCU is “in fault state” or when the RCC indicated the switch of the system clock on safe clock. The enabling of HRTIM system fault individual sources is SW programmable by the system configuration controller (SYSCFG) in the HRTIMxFLT_TIMxBRK_EN register.

There are 2 sets of fault inputs, refer to *Table 31: HRTIM1 fault inputs (protection)* for HRTIM1 and *Table 32: HRTIM2 fault inputs (protection)* for HRTIM2.

Table 31. HRTIM1 fault inputs (protection)

Fault channel	External Input FLTxSRC[1:0] = 00	On-chip source FLTxSRC[1:0] = 01	External Input FLTxSRC[1:0] = 10	On-chip source FLTxSRC[1:0] = 11
hrtim_flt1[4:1]	HRTIM_FLT1 (PA8/PI7)	Comparator output (COMP2)	EEV1_muxout	HRTIM2_EEV1_muxout
hrtim_flt2[4:1]	HRTIM_FLT2 (PB1/PI5)	Comparator output (COMP4)	EEV2_muxout	HRTIM2_EEV2_muxout
hrtim_flt3[4:1]	HRTIM_FLT3 (PI3)	Comparator output (COMP6)	EEV3_muxout	HRTIM2_EEV3_muxout
hrtim_flt4[4:1]	HRTIM_FLT4 (PI1)	Comparator output (COMP1)	EEV4_muxout	HRTIM2_EEV4_muxout
hrtim_flt5[4:1]	HRTIM_FLT5 (PI9)	Comparator output (COMP3)	EEV5_muxout	HRTIM2_EEV5_muxout
hrtim_flt6[4:1]	HRTIM_FLT6 (PI8)	Comparator output (COMP5)	EEV6_muxout	HRTIM2_EEV6_muxout

Table 32. HRTIM2 fault inputs (protection)

Fault channel	External Input FLTxSRC[1:0] = 00	On-chip source FLTxSRC[1:0] = 01	External Input FLTxSRC[1:0] = 10	On-chip source FLTxSRC[1:0] = 11
hrtim_flt1[4:1]	HRTIM_FLT1 (PD5)	Comparator output (COMP1)	EEV1_muxout	HRTIM1_EEV1_muxout
hrtim_flt2[4:1]	HRTIM_FLT2 (PD3)	Comparator output (COMP3)	EEV2_muxout	HRTIM1_EEV2_muxout
hrtim_flt3[4:1]	HRTIM_FLT3 (PD0)	Comparator output (COMP5)	EEV3_muxout	HRTIM1_EEV3_muxout
hrtim_flt4[4:1]	HRTIM_FLT4 (PC14)	Comparator output (COMP2)	EEV4_muxout	HRTIM1_EEV4_muxout
hrtim_flt5[4:1]	HRTIM_FLT5 (PD2)	Comparator output (COMP4)	EEV5_muxout	HRTIM1_EEV5_muxout
hrtim_flt6[4:1]	HRTIM_FLT6 (PH10)	Comparator output (COMP6)	EEV6_muxout	HRTIM1_EEV6_muxout

5.5.1.3 HRTIM DAC trigger connections

Table 33. HRTIM1 DAC trigger connections

HRTIM DAC triggers	DAC1_CH1 DAC1_CH2	DAC2_CH1 DAC2_CH2	DAC3_CH1 DAC3_CH2	DAC4_CH1 DAC4_CH2
hrtim_dac_trg1	Yes	—	—	—
hrtim_dac_trig2	—	Yes	—	—
hrtim_dac_reset_trg_A hrtim_dac_step_trg_A	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_B hrtim_dac_step_trg_B	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_C hrtim_dac_step_trg_C	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_D hrtim_dac_step_trg_D	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_E hrtim_dac_step_trg_E	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_F hrtim_dac_step_trg_F	Yes	Yes	Yes	Yes

Table 34. HRTIM2 DAC trigger connections

HRTIM DAC triggers	DAC1_CH1 DAC1_CH2	DAC2_CH1 DAC2_CH2	DAC3_CH1 DAC3_CH2	DAC4_CH1 DAC4_CH2
hrtim_dac_trg1	—	—	Yes	—
hrtim_dac_trig2	—	—	—	Yes
hrtim_dac_reset_trg_A hrtim_dac_step_trg_A	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_B hrtim_dac_step_trg_B	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_C hrtim_dac_step_trg_C	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_D hrtim_dac_step_trg_D	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_E hrtim_dac_step_trg_E	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg_F hrtim_dac_step_trg_F	Yes	Yes	Yes	Yes

5.5.1.4 HRTIM update and synchronization signals

Table 35. HRTIMx burst mode (x = {1; 2})

HRTIMx Burst mode trigger event/ clock signal name	HRTIM Burst mode trigger event/ clock signal assignment
hrtimx_bm_trg	tim7_trgo
hrtimx_bm_ck1	tim16_oc1
hrtimx_bm_ck2	Reserved
hrtimx_bm_ck3	tim7_trgo
hrtimx_bm_ck4	Reserved

Table 36. HRTIMx update enable signals (x = {1; 2})

HRTIM update enable signal	HRTIM update enable assignment
hrtim_upd_en1	tim16_oc1
hrtim_upd_en2	SW control, see HRTIMx_UPDATE_EN in SYSCFG chapter
hrtim_upd_en3	tim6_trgo

Table 37. HRTIM1 synchronization input signals

HRTIM1 synchronization input	HRTIM1 synchronization signal source
hrtim1_in_sync1	Reserved (not used)
hrtim1_in_sync2	tim1_trgo
hrtim1_in_sync3	HRTIM_SCIN pin (pad PA3)

Table 38. HRTIM1 synchronization output signals

HRTIM1 synchronization output	HRTIM1 synchronization output destination
hrtim1_out_sync[1]	hrtim1_scout1 (internal signal to HRTIM2)
hrtim1_out_sync[2]	HRTIM1_SCOUT (pad PA2) hrtim1_out_sync2 (internal signal to TIMx)

Table 39. HRTIM2 synchronization input signals

HRTIM2 synchronization input	HRTIM2 synchronization signal source
hrtim2_in_sync1	hrtim1_out_sync[1] (aka hrtim1_scout1)
hrtim2_in_sync2	tim1_trgo
hrtim2_in_sync3	HRTIM_SCIN (pad PA3)

Table 40. HRTIM2 synchronization output signals

HRTIM2 synchronization output	HRTIM2 synchronization output destination
hrtim2_out_sync[1]	Reserved (not used)
hrtim2_out_sync[2]	hrtim2_out_sync2 (internal signal to TIMx)

5.5.2 TIMx timer overview

The following section gives an overview of the features implemented in the various TIMx Timer modules.

TIMx instances with x=2, 3, 6, 7, TS uses twice the APB1 clock as reference (that is half of the system clock).

TIMx instances with x=1, 8, 4, 5, 15, 16 uses twice the APB2 clock as reference (that is the system clock).

Table 41. TIMX instance features overview

Feature	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM15	TIM16
	Adv Control		General Purpose					
Resolution (bits)	16	16	32	16	16	32	16	16
Prescaler	16-bit							
Counter direction	Up, Down, Up&Down		Up, Down, Up&Down				Up	
Number of channels	6: – CH1/1N..CH4/4N – CH5 & CH6 output only (not on pads)		4: – CH1..CH4				2: – CH1/1N – CH2	1: – CH1/1N
Input capture mode	Yes		Yes				Yes	
PWM input mode	Yes		Yes				Yes	No
Forced output mode	Yes		Yes				Yes	
Output compare mode	Yes		Yes				Yes	
PWM modes	Standard Asymmetric Combined Combined 3-phase 6-step PWM		Standard Asymmetric Combined				Standard Asymm. Combined	Standard
PWM dithering	Yes		Yes				Yes	

Table 41. TIMX instance features overview (continued)

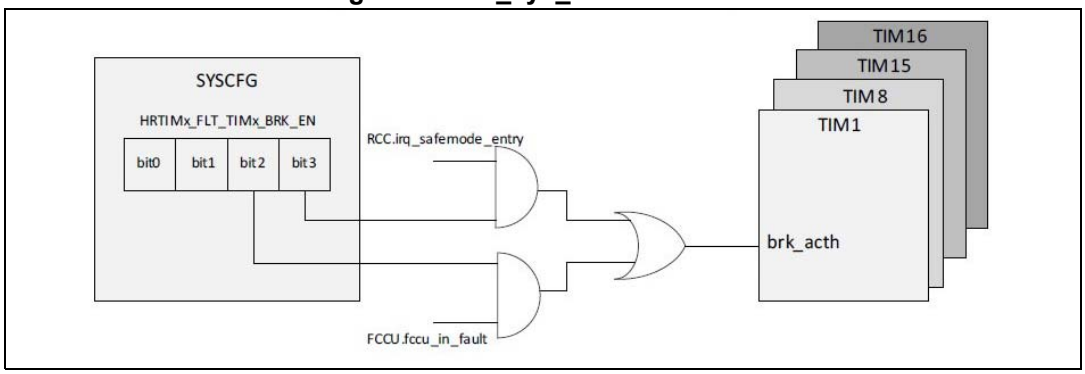
Feature	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM15	TIM16
	Adv Control		General Purpose					
Prog. dead-time	Yes: CH1..CH4		No				Yes: CH1	
Break inputs	2x Bidirectional		No				1x Bidirectional	
One-Pulse Mode	Yes		Yes				Yes	
Retrig. One pulse mode	Yes		Yes				Yes	No
Encoder interface mode	Quadrature Clock plus direction Directional clock		Quadrature Clock plus direction Directional clock				No	No
Index input	Yes		Yes				No	
Encoder error management	Yes		Yes				No	
Timer input XOR function	Yes		Yes				Yes	No
DMA	Yes		Yes				Yes	

Table 42. TIMx Basic & timestamp timer features overview

Feature	TIM6	TIM7	TIM_TS
Resolution (bits)	16		32
Prescaler	16-bit		16-bit
Counter direction	Up		Up
Number of channels	0		0
DMA	Yes		Yes

5.5.2.1 TIMx timer fault inputs

Figure 11. tim_sys_brk connection



Each TIM has a system break input, tim_sys_brk, that allows to put the timer in a safe state, when asserted.

This input is connected so that an internal critical device fault brings the TIM outputs to a safe state. This is achieved by connecting to TIM break fault the indication that FCCU is “in fault state” and the indication from RCC that the system clock has switched to the safe clock.

The enabling of TIM break fault individual sources is software programmable by the system configuration controller (SYSCFG) in the HRTIMxFLT_TIMxBRK_EN register, as shown in *Figure 11: tim_sys_brk connection*.

This connection applies to TIM1/TIM8/TIM15/TIM16 instances.

5.5.3 TIM1/TIM8 configuration

5.5.3.1 Interconnects to the tim_ti[1..4] input multiplexers

Table 43. Interconnect to the TIMx tim_ti1 input multiplexer (x=1, 8)

tim_ti1 input	Sources	
	TIM1	TIM8
tim_ti1_in0	TIM1_CH1	TIM8_CH1
tim_ti1_in1	comp1_out	comp1_out
tim_ti1_in2	comp2_out	comp2_out
tim_ti1_in3	comp3_out	comp3_out
tim_ti1_in4	comp4_out	comp4_out
tim_ti1_in5	sdadc1_watchdog	sdadc1_watchdog
tim_ti1_in6	sdadc2_watchdog	sdadc2_watchdog
tim_ti1_in7	sdadc1_data_available	sdadc1_data_available
tim_ti1_in8	sdadc2_data_available	sdadc2_data_available
tim_ti1_in[9..15]	Reserved	Reserved

Table 44. Interconnect to the TIMx tim_ti2 input multiplexer (x=1, 8)

tim_ti1 input	Sources	
	TIM1	TIM8
tim_ti2_in0	TIM1_CH2	TIM8_CH2
tim_ti2_in[1..15]	Reserved	Reserved

Table 45. Interconnect to the TIMx tim_ti3 input multiplexer (x=1, 8)

tim_ti1 input	Sources	
	TIM1	TIM8
tim_ti3_in0	TIM1_CH3	TIM8_CH3
tim_ti3_in1	sdadc1_watchdog	sdadc1_watchdog
tim_ti3_in2	sdadc2_watchdog	sdadc2_watchdog
tim_ti3_in3	sdadc1_data_available	sdadc1_data_available
tim_ti3_in4	sdadc2_data_available	sdadc2_data_available
tim_ti3_in[5..15]	Reserved	Reserved

Table 46. Interconnect to the TIMx tim_ti4 input multiplexer (x=1, 8)

tim_ti1 input	Sources	
	TIM1	TIM8
tim_ti4_in0	TIM1_CH4	TIM8_CH4
tim_ti4_in1	sdadc1_data_available	sdadc1_data_available
tim_ti4_in2	sdadc2_data_available	sdadc2_data_available
tim_ti4_in[3..15]	Reserved	Reserved

5.5.3.2 Interconnects to the trigger input multiplexers

Table 47. TIMx internal trigger connection (x=1, 8)

TIMx	TIM1	TIM8
tim_itr0	Reserved	tim1_trgo
tim_itr1	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo
tim_itr5	tim8_trgo	Reserved
tim_itr6	tim15_trgo	tim15_trgo
tim_itr7	tim16_oc1	tim16_oc1

Table 47. TIMx internal trigger connection (x=1, 8) (continued)

TIMx	TIM1	TIM8
tim_itr8	Reserved	Reserved
tim_itr9	hrtim2_out_scout2	hrtim2_out_scout2
tim_itr10	hrtim1_out_scout2	hrtim1_out_scout2
tim_itr[11..15]	Reserved	Reserved

Table 48. TIMx external trigger connection (x=1, 8)

Timer external trigger input signal	Timer external trigger signals assignment	
	TIM1	TIM8
tim_etr0	TIM1_ETR	TIM8_ETR
tim_etr1	comp1_out	comp1_out
tim_etr2	comp2_out	comp2_out
tim_etr3	comp3_out	comp3_out
tim_etr4	comp4_out	comp4_out
tim_etr5	comp5_out	comp5_out
tim_etr6	comp6_out	comp6_out
tim_etr7	comp7_out	comp7_out
tim_etr8	comp8_out	comp8_out
tim_etr9	adc1_awd1	adc2_awd1
tim_etr10	adc1_awd2	adc2_awd2
tim_etr11	adc1_awd3	adc2_awd3
tim_etr12	adc4_awd1	adc3_awd1
tim_etr13	adc4_awd2	adc3_awd2
tim_etr14	adc4_awd3	adc3_awd3
tim_etr15	Reserved	Reserved

5.5.3.3 Interconnects to the break & break2 inputs

Table 49. TIMx break interconnect (x=1, 8)

tim_brk input	TIM1	TIM8
TIMx_BKIN	TIM1_BKIN pin	TIM8_BKIN pin
tim_brk_cmp1	comp1_out	comp1_out
tim_brk_cmp2	comp2_out	comp2_out
tim_brk_cmp3	comp3_out	comp3_out
tim_brk_cmp4	comp4_out	comp4_out
tim_brk_cmp5	comp5_out	comp5_out

Table 49. TIMx break interconnect (x=1, 8) (continued)

tim_brk input	TIM1	TIM8
tim_brk_cmp6	comp6_out	comp6_out
tim_brk_cmp7	comp7_out	comp7_out
tim_brk_cmp8	comp8_out	comp8_out

Table 50. TIMx break2 interconnect (x=1, 8)

tim_brk input	TIM1	TIM8
TIMx_BKIN2	TIM1_BKIN2 pin	TIM8_BKIN2 pin
tim_brk2_cmp1	comp1_out	comp1_out
tim_brk2_cmp2	comp2_out	comp2_out
tim_brk2_cmp3	comp3_out	comp3_out
tim_brk2_cmp4	comp4_out	comp4_out
tim_brk2_cmp5	comp5_out	comp5_out
tim_brk2_cmp6	comp6_out	comp6_out
tim_brk2_cmp7	comp7_out	comp7_out
tim_brk2_cmp8	comp8_out	comp8_out

5.5.3.4 Interconnects to the ocref_clr input multiplexer

Table 51. Interconnect to the TIMx ocref_clr input multiplexer (x=1, 8)

Timer OCREF clear signal	Timer OCREF clear signals assignment	
	TIM1	TIM8
tim_ocref_clr0	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out
tim_ocref_clr2	comp3_out	comp3_out
tim_ocref_clr3	comp4_out	comp4_out
tim_ocref_clr4	comp5_out	comp5_out
tim_ocref_clr5	comp6_out	comp6_out
tim_ocref_clr6	comp7_out	comp7_out
tim_ocref_clr7	—	—

5.5.4 TIM2/TIM3/TIM4/TIM5 configuration

5.5.4.1 Interconnects to the tim_ti[1..4] input multiplexers

Table 52. Interconnect to the TIMx tim_ti1 input multiplexer (x=2..5)

tim_ti1 input	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti1_in0	TIM2 external TI1 input pins	TIM3 external TI1 input pins	TIM4 external TI1 input pins	TIM5 external TI1 input pins
tim_ti1_in1	comp1_out	comp1_out	comp1_out	LSI
tim_ti1_in2	comp2_out	comp2_out	comp2_out	Reserved
tim_ti1_in3	comp3_out	comp3_out	comp3_out	comp1_out
tim_ti1_in4	comp4_out	comp4_out	comp4_out	comp2_out
tim_ti1_in5	comp5_out	comp5_out	comp5_out	comp3_out
tim_ti1_in6	sdadc1_watchdog	comp6_out	comp6_out	comp4_out
tim_ti1_in7	sdadc2_watchdog	comp7_out	comp7_out	comp5_out
tim_ti1_in8	sdadc1_data_ available	comp8_out	comp8_out	comp6_out
tim_ti1_in9	sdadc2_data_ available	sdadc1_watchdog	sdadc1_watchdog	comp7_out
tim_ti1_in10	Reserved	sdadc2_watchdog	sdadc2_watchdog	comp8_out
tim_ti1_in[11..15]	Reserved	Reserved	Reserved	Reserved

Table 53. Interconnect to the TIMx tim_ti2 input multiplexer (x=2..5)

tim_ti2 input	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in0	TIM2 external TI2 input pins	TIM3 external TI2 input pins	TIM4 external TI2 input pins	TIM5 external TI2 input pins
tim_ti2_in1	comp1_out	comp1_out	comp1_out	comp1_out
tim_ti2_in2	comp2_out	comp2_out	comp2_out	comp2_out
tim_ti2_in3	comp3_out	comp3_out	comp3_out	comp3_out
tim_ti2_in4	comp4_out	comp4_out	comp4_out	comp4_out
tim_ti2_in5	comp5_out	comp5_out	comp5_out	comp5_out
tim_ti2_in6	comp6_out	comp6_out	comp6_out	comp6_out
tim_ti2_in7	comp7_out	comp7_out	comp7_out	comp7_out
tim_ti2_in8	comp8_out	comp8_out	comp8_out	comp8_out
tim_ti2_in[9..15]	Reserved	Reserved	Reserved	Reserved

Table 54. Interconnect to the TIMx tim_ti3 input multiplexer (x=2..5)

tim_ti3 input	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti3_in0	TIM2 external TI3 input pins	TIM3 external TI3 input pins	TIM4 external TI3 input pins	TIM5 external TI3 input pins
tim_ti3_in1	comp4_out	comp3_out	comp5_out	sdadc1_watchdog
tim_ti3_in2	sdadc1_watchdog	sdadc1_watchdog	sdadc1_watchdog	sdadc2_watchdog
tim_ti3_in3	sdadc2_watchdog	sdadc2_watchdog	sdadc2_watchdog	sdadc1_data_available
tim_ti3_in4	sdadc1_data_available	sdadc1_data_available	sdadc1_data_available	sdadc2_data_available
tim_ti3_in5	sdadc2_data_available	sdadc2_data_available	sdadc2_data_available	Reserved
tim_ti3_in [6..15]	Reserved	—	Reserved	Reserved

Table 55. Interconnect to the TIMx tim_ti4 input multiplexer (x=2..5)

tim_ti4 input	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti4_in0	TIM2 external TI4 input pins	TIM3 external TI4 input pins	TIM4 external TI4 input pins	TIM5 external TI4 input pins
tim_ti4_in1	comp1_out	sdadc1_data_available	comp6_out	sdadc1_data_available
tim_ti4_in2	comp2_out	sdadc2_data_available	sdadc1_watchdog	sdadc2_data_available
tim_ti4_in3	sdadc1_watchdog	Reserved	sdadc2_watchdog	Reserved
tim_ti4_in4	sdadc2_watchdog	Reserved	sdadc1_data_available	Reserved
tim_ti4_in5	sdadc1_data_available	Reserved	sdadc2_data_available	Reserved
tim_ti4_in6	sdadc2_data_available	Reserved	Reserved	Reserved
tim_ti4_in [7..15]	Reserved	Reserved	Reserved	Reserved

5.5.4.2 Interconnects to the trigger input multiplexers

Table 56. TIMx internal trigger connection (x=2..5)

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr0	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_itr1	Reserved	tim2_trgo	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	Reserved	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo	Reserved	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo	tim5_trgo	Reserved
tim_itr5	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_itr6	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo

Table 56. TIMx internal trigger connection (x=2..5) (continued)

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr7	tim16_oc1	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr8	Reserved	Reserved	Reserved	Reserved
tim_itr9	hrtim2_out_scout2	hrtim2_out_scout2	hrtim2_out_scout2	hrtim2_out_scout2
tim_itr10	hrtim1_out_scout2	hrtim1_out_scout2	hrtim1_out_scout2	hrtim1_out_scout2
tim_itr[11..15]	Reserved	Reserved	Reserved	Reserved

Table 57. TIMx external trigger connection (x=2..5)

Timer external trigger input signal	Timer external trigger signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR
tim_etr1	comp1_out	comp1_out	comp1_out	comp1_out
tim_etr2	comp2_out	comp2_out	comp2_out	comp2_out
tim_etr3	comp3_out	comp3_out	comp3_out	comp3_out
tim_etr4	comp4_out	comp4_out	comp4_out	comp4_out
tim_etr5	comp5_out	comp5_out	comp5_out	comp5_out
tim_etr6	comp6_out	comp6_out	comp6_out	comp6_out
tim_etr7	comp7_out	comp7_out	comp7_out	comp7_out
tim_etr8	comp8_out	comp8_out	comp8_out	comp8_out
tim_etr9	tim3_etr	tim2_etr	tim3_etr	tim2_etr
tim_etr10	tim4_etr	tim4_etr	tim5_etr	tim3_etr
tim_etr11	tim5_etr	Reserved	adc1_awd1	adc1_awd1
tim_etr12	adc1_awd1	adc2_awd1	adc2_awd1	adc2_awd1
tim_etr13	adc2_awd1	adc2_awd2	adc3_awd1	adc3_awd1
tim_etr14	adc3_awd1	adc2_awd3	adc4_awd1	adc4_awd1
tim_etr15	adc4_awd1	Reserved	adc5_awd1	adc5_awd1

5.5.4.3 Interconnects to the ocref_clr input multiplexer

Table 58. Interconnect to the TIMx ocref_clr input multiplexer (x=2..5)

Timer OCREF clear signal	Timer OCREF clear signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr0	comp1_out	comp1_out	Reserved	Reserved
tim_ocref_clr1	comp2_out	comp2_out	Reserved	Reserved
tim_ocref_clr2	comp3_out	comp3_out	Reserved	Reserved
tim_ocref_clr3	comp4_out	comp4_out	Reserved	Reserved

Table 58. Interconnect to the TIMx ocref_clr input multiplexer (x=2..5) (continued)

Timer OCREF clear signal	Timer OCREF clear signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr4	comp5_out	comp5_out	Reserved	Reserved
tim_ocref_clr5	comp6_out	comp6_out	Reserved	Reserved
tim_ocref_clr6	comp7_out	comp7_out	Reserved	Reserved
tim_ocref_clr7	—	—	Reserved	Reserved

5.5.5 TIM15/TIM16 configuration

5.5.5.1 Interconnects to the tim_ti[1..4] input multiplexers

Table 59. Interconnect to the TIMx tim_ti1 input multiplexer (x=15, 16)

tim_ti1 input	Sources	
	TIM15	TIM16
tim_ti1_in0	TIM15 external TI1 input pins	TIM16 external TI1 input pins
tim_ti1_in1	Reserved	comp6_out
tim_ti1_in2	comp1_out	MCO
tim_ti1_in3	comp2_out	HSE_Div32
tim_ti1_in4	comp5_out	Reserved
tim_ti1_in5	comp7_out	LSI
tim_ti1_in6	sdadc1_data_available	sdadc1_watchdog
tim_ti1_in7	sdadc2_data_available	sdadc2_watchdog
tim_ti1_in8	Reserved	sdadc1_data_available
tim_ti1_in9	Reserved	sdadc2_data_available
tim_ti1_in[10..15]	Reserved	Reserved

Table 60. Interconnect to the TIMx tim_ti2 input multiplexer (x=15)

tim_ti1 input	Source
	TIM15
tim_ti2_in0	TIM15 external TI2 input pins
tim_ti2_in1	comp2_out
tim_ti2_in2	comp3_out
tim_ti2_in3	comp6_out
tim_ti2_in4	comp7_out
tim_ti2_in5	sdadc1_watchdog
tim_ti2_in6	sdadc2_watchdog

Table 60. Interconnect to the TIMx tim_ti2 input multiplexer (x=15) (continued)

tim_ti1 input	Source
	TIM15
tim_ti2_in7	sdadc1_data_available
tim_ti2_in8	sdadc2_data_available
tim_ti2_in[9..15]	Reserved

5.5.5.2 Interconnects to the trigger input multiplexers

Table 61. TIM15 internal trigger connection

TIMx	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	tim5_trgo
tim_itr5	tim8_trgo
tim_itr6	Reserved
tim_itr7	tim16_oc1
tim_itr8	Reserved
tim_itr9	hrtim2_out_scout2
tim_itr10	hrtim1_out_scout2
tim_itr[11..15]	Reserved

5.5.5.3 Interconnects to the break inputs

Table 62. TIMx break interconnect (x=15, 16)

tim_brk input	TIM15	TIM16
TIMx_BKIN	TIM15_BKIN pin	TIM16_BKIN pin
tim_brk_cmp1	comp1_out	comp1_out
tim_brk_cmp2	comp2_out	comp2_out
tim_brk_cmp3	comp3_out	comp3_out
tim_brk_cmp4	comp4_out	comp4_out
tim_brk_cmp5	comp5_out	comp5_out
tim_brk_cmp6	comp6_out	comp6_out
tim_brk_cmp7	comp7_out	comp7_out
tim_brk_cmp8	comp8_out	comp8_out

5.5.5.4 Interconnects to the ocref_clr input multiplexer

Table 63. Interconnect to the TIMx ocref_clr input multiplexer (x=15, 16)

Timer OCREF clear signal	Timer OCREF clear signals assignment	
	TIM15	TIM16
tim_ocref_clr0	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out
tim_ocref_clr2	comp3_out	comp3_out
tim_ocref_clr3	comp4_out	comp4_out
tim_ocref_clr4	comp5_out	comp5_out
tim_ocref_clr5	comp6_out	comp6_out
tim_ocref_clr6	comp7_out	comp7_out
tim_ocref_clr7	—	—

5.5.6 Timestamp timer configuration

The SR5E1x features one 32-bit timestamp timer (TIM_TS).

TIM_TS generates timestamp information for the SDADC and CAN modules.

TIM_TS is clocked by the APB1 clock (PCLK1).

5.5.7 Independent watchdog (IWDG)

The SR5E1x includes two IWDG: IWDG1 and IWDG2.

Both IWDG instances are clocked by the internal RC Oscillator (IRCOSC, 16 MHz) and by default disabled.

They can be enabled by hardware after reset via the bits IWDG1_HW and IWDG2_HW of the UTEST Miscellaneous DCF record.

5.5.8 System window watchdog (WWDG)

The SR5E1x includes two WWDG: WWDG1 and WWDG2.

Both instances are clocked at the APB1 bridge frequency.

The WWDGs can trigger an early warning interrupt (WWDG1 to Core1, WWDG2 to Core2).

5.5.9 RTC configuration

The real-time clock (RTC) is an independent Binary-Coded Decimal (BCD) timer/counter.

It can be used for counting long time intervals, but it cannot be used as real RTC as data is not maintained in an ALWAYS_ON domain.

The RTC clock source is configured by the Reset and clock control module (RCC). The possible clock sources are:

- The internal low-speed RC oscillator (LSI), with a configurable prescaler
- The external clock (XOSC), with a configurable prescaler

The RTC events (Alarm, WakeUp Timer) can generate an interrupt, passing through the EXTI module.

5.6 Communication interfaces

5.6.1 CAN subsystem configuration

For the CAN subsystem configuration refer to *Chapter 46: CAN subsystem*.

5.6.2 SPI configuration

The SR5E1x contains four Serial peripheral interface (SPI/I2S) modules.

The instances SPI2 and SPI3 feature in addition to the / inter-IC sound (I2S) protocol.

SPI1 and SPI4 are connected on the APB2 peripheral bridge and clocked by PCLK2 clock.

SPI2 and SPI3 are connected on the APB1 peripheral bridge and clocked by PCLK1 clock.

5.6.3 UART configuration

The SR5E1x contains three Universal asynchronous receiver transmitter (UART) modules, supporting the local interconnection network (LIN) protocol.

Each UART receives two clock signals: a kernel clock and a register interface clock.

For all instances, the kernel clock is the UART_CLK provided by the RCC.

UART1 is connected on the APB2 peripheral bridge and its register interface is clocked by the PCLK2 clock.

UART2 and UART3 are connected on the APB1 peripheral bridge and their register interfaces are clocked by PCLK1 clock.

5.6.4 I2C configuration

The SR5E1x contains two inter-integrated circuits (I2C).

I2C1 and I2C2 are connected on the APB1 peripheral bridge. They receive two clocks:

- I2CCLK the kernel clock from the RCC (configurable between PLL0, XOSC, and IRCOSC).
- APB1 clock (PCLK1) for register access.

5.7 Reset and boot modules

5.7.1 Boot assist firmware (BAF) configuration

The BAF is executed out of reset by the boot core (Core1). BAF uses the following resources.

Table 64. RAM used by the BAF

Start	End	Description
0x2000_0000	0x2000_068F	Boot CPU (Core1) DMEM: used for stack and variables

Table 65. Pins used by the BAF for serial boot mode

Pin name	Pad configuration	Function
PA[4]	Input, pull-up	Rx signal for CAN1, configured at start of serial boot mode
PA[5]	Output, push-pull mode, medium speed	Tx signal for CAN1, configured after reception of a valid message
PF[2]	Input, pull-up	Rx signal for UART1, configured at start of serial boot mode
PF[3]	Output, push-pull mode, medium speed	Tx signal for UART1, configured after reception of a valid message on UART1

5.7.2 System status and configuration module (SSCM) configuration

This section summarizes the SSCM configuration in the device. For a comprehensive description of the SSCM, please refer to the SSCM's dedicated chapter.

5.7.2.1 SSCM instantiation

There is one SSCM instance in the device. The SSCM is part of the reset and boot subsystem and is clocked by the IRCOSC.

The SSCM reads the DCF records from the Flash in order to configure the reset vector for the Boot CPU and the information to start the other CPUs.

5.7.2.2 Device-specific features

The SSCM allows Boot Assist from Flash (BAF), and utilizes PASS security concept.

Table 66 shows the address locations that the SSCM uses to search for startup information.

Table 66. SSCM startup information addresses

Value	Description
0x1FF8_0000	The location at which the SSCM looks for the UTEST DCF records
0x1FF0_0000	Location of the BAF in flash

5.8 Safety modules

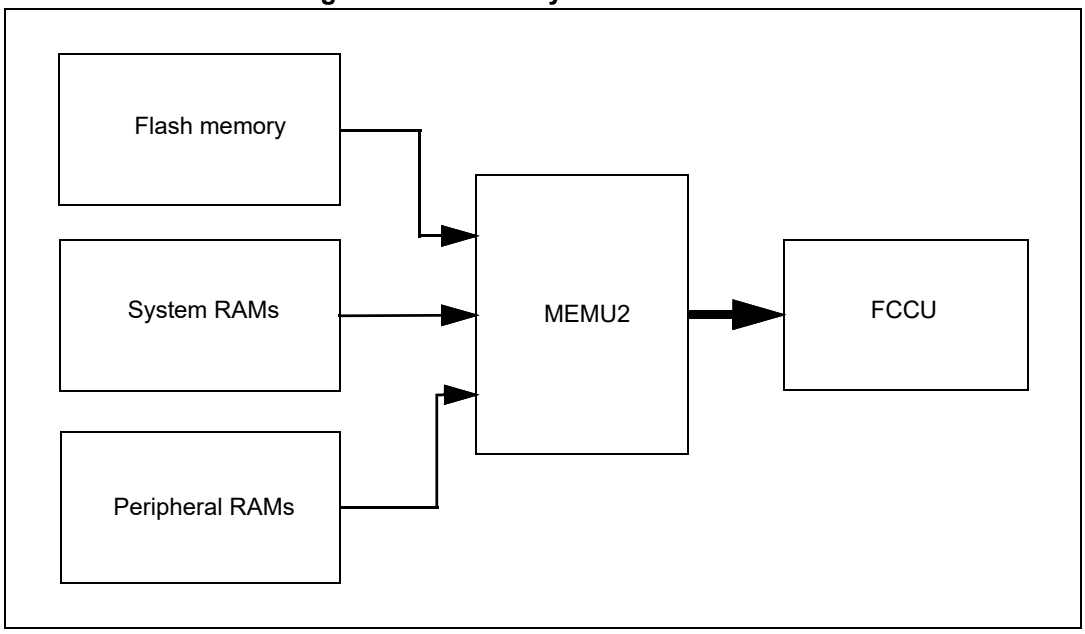
5.8.1 Cyclic redundancy check (CRC)

The device embeds one CRC module. The CRC is connected on the AHB1 bridge and receives the AHB clock (system clock).

5.8.2 Memory error management unit (MEMU2)

5.8.2.1 MEMU2 system connections

Figure 12. MEMU2 system connections



5.8.2.2 MEMU2 error sources

Table 67 shows the MEMU error sources for each instance configured on this device.

Table 67. MEMU2 error sources

Instance	Value
Number of system RAM unique error sources	45
Number of peripheral RAM unique error sources	38
Number of flash memory unique error sources	1

5.8.2.3 Reporting tables implementation

There are 3 categories of error reporting for the MEMU2:

- Flash error reporting
- System RAMs error reporting
- Peripheral RAM error reporting

Each category of error reporting has two tables associated with it, these are:

- Correctable error reporting table
- Uncorrectable error reporting table

Each entry in a reporting table corresponds to a unique error event. The number of entries supported by each table depends upon the error source.

The table below describes the number of entries in each reporting table for the different error sources.

Table 68. Number of entries for each MEMU reporting table

Error source	Entries in single correctable error reporting table	Entries in double correctable error reporting table	Entries in uncorrectable error reporting table
Flash	32	1	1
System RAMs	10	—	1
Peripheral RAMs	2	—	1

Table 69. Corresponding error sources for system RAM OFLW0 registers

Source number	SYS_RAM_OFLWn bit field position (bit)	Error source
0	0	SRAM1/SRAM2
1	1	Cortex®-M7_1 Instruction Cache
2	2	Cortex®-M7_1 Data Cache
3	3	Cortex®-M7_1 ITCM
4	4	Cortex®-M7_1 DTCM
5	5	Cortex®-M7_2 Instruction Cache
6	6	Cortex®-M7_2 Data Cache
7	7	Cortex®-M7_2 ITCM
8	8	Cortex®-M7_2 DTCM
Reserved		
32	32	MBIST Cortex®-M7 Core 1 Instruction Cache
33	33	MBIST Cortex®-M7_1 Data Cache 0
34	34	MBIST Cortex®-M7_1 Data Cache 1
35	35	MBIST Cortex®-M7_1 Data Cache tag
36	36	MBIST Cortex®-M7_1 ITCM
37	37	MBIST Cortex®-M7_1 DTCM
38	38	MBIST Cortex®-M7 Core 2 Instruction Cache
39	39	MBIST Cortex®-M7_2 Data Cache 0
40	40	MBIST Cortex®-M7_2 Data Cache 1
41	41	MBIST Cortex®-M7_2 Data Cache tag
42	42	MBIST Cortex®-M7_2 ITCM
43	43	MBIST Cortex®-M7_2 DTCM
44	44	MBIST SRAM1/SRAM2

Table 70. Corresponding error sources for PERIPH_RAM_OFLWn registers

Source number	PERIPH_RAM_OFLWn bit field position (bit)	Error source
0	0	CAN RAM
1	1	HSM RAM
2	2	HSM SBI Cache (Instruction Cache)
Reserved		
32	32	MBIST CAN RAM
33	33	Flash RAM
34	34	HSM PRAM
35	35	HSM SBI Cache
36	36	HSM MPAES
37	37	ETF RAM

Table 71. Corresponding error sources for NVM_OFLWn registers

Source number	NVM_OFLWn bit field position (bit)	Error source
0	0	NVMC1 for RWR0 - Code correctable (single) NVMC2 for RWR1 - Code correctable (single)

5.8.3 Indirect memory access (IMA) configuration

The IMA refers to the alteration of memory data values during system development and test activities. This section provides details about what memory locations can be accessed, either via processor core access or via the IMA module, and how they are configured.

For more details, refer to *Chapter 54: Indirect memory access (IMA)*.

5.8.3.1 Memory accessible via the IMA modules

The SRAM arrays are accessible via the IMA modules. Refer to the table below.

Table 72. Memory arrays accessible via IMA

IMA array select	Block	Function	Words	Data bits	ECC bits
0	None	IMA disabled	—	—	—

Table 72. Memory arrays accessible via IMA (continued)

IMA array select	Block	Function	Words	Data bits	ECC bits
1	Core1	ICache_0	512	64	8
2		ICache_1	512	64	8
3		ICache_Tag_0	128	21	7
4		ICache_Tag_1	128	21	7
5		DCache_0_0	512	32	7
6		DCache_0_1	512	32	7
7		DCache_0_2	512	32	7
8		DCache_0_3	512	32	7
9		DCache_1_0	512	32	7
10		DCache_1_1	512	32	7
11		DCache_1_2	512	32	7
12		DCache_1_3	512	32	7
13		DCache_Tag0	128	24	7
14		DCache_Tag1	128	24	7
15		DCache_Tag2	128	24	7
16		DCache_Tag3	128	24	7
17		ITCM	4096	64	8
18		DTCM array A(1)	4096	32	7
19		DTCM array B(2)	4096	32	7
20		DTCM array C(3)	4096	32	7
21		DTCM array D(4)	4096	32	7

Table 72. Memory arrays accessible via IMA (continued)

IMA array select	Block	Function	Words	Data bits	ECC bits
22	Core2	ICache_0	512	64	8
23		ICache_1	512	64	8
24		ICache_Tag_0	128	21	7
25		ICache_Tag_1	128	21	7
26		DCache_0_0	512	32	7
27		DCache_0_1	512	32	7
28		DCache_0_2	512	32	7
29		DCache_0_3	512	32	7
30		DCache_1_0	512	32	7
31		DCache_1_1	512	32	7
32		DCache_1_2	512	32	7
33		DCache_1_3	512	32	7
34		DCache_Tag0	128	24	7
35		DCache_Tag1	128	24	7
36		DCache_Tag2	128	24	7
37		DCache_Tag3	128	24	7
38		ITCM	4096	64	8
39		DTCM array A(1)	4096	32	7
40		DTCM array B(2)	4096	32	7
41		DTCM array C(3)	4096	32	7
42		DTCM array D(4)	4096	32	7
43	CAN	CAN Message RAM	4096	32	7

1. Gives the access to the first 32-bit word of the first 32 kilobytes block, that is offset 0x0 - 0x8 - 0x10, and so on.
2. Gives the access to the first 32-bit word of the last 32 kilobytes block, that is offset 0x8000 - 0x8008 - 0x8010, and so on.
3. Gives the access to the second 32-bit word of the first 32 kilobytes block, that is offset 0x4 - 0xC - 0x14, and so on.
4. Gives the access to the second 32-bit word of the last 32 kilobytes block, that is offset 0x8004 - 0x800C - 0x8014, and so on.

5.8.4 Fault collection & control unit (FCCU) configuration

5.8.4.1 Available FCCU output signals

The following table shows the available FCCU output signals implemented on this chip.

Table 73. FCCU input, output signals

FCCU signal name	Description	Device signal name
EOUT[0], FI[0], ERROR0 F0	bidirectional signal, can be configured as output only EOUT0	EOUT0 (PA9)
EOUT[1], FI[1], ERROR1 F1	bidirectional signal, can be configured as output only EOUT1	EOUT1 (PA10)
EIN[0]	Unidirectional input error signal	EIN0 (PD9)
EIN[1]	Unidirectional input error signal	EIN1 (PH13)

5.8.4.1.1 External error indication

Failure of the MCU is signaled to one or two pins, FCCU_ERR0 (PA[9]) and FCCU_ERR1 (PA[10]). External failure to MCU is signaled from FCCU_EIN0 (PA[9] or PD[9]) or FCCU_EIN1 (PA[10] or PH[13]) (refer to “Fault collection and control unit (FCCU)” chapter, and “FCCU configuration” section of the “Device configuration” chapter, for details on the fault output signals). PA[9] and PA[10] pins are dedicated to FCCU while the pins PD[9] and PH[13] can be multiplexed with other functions with different values.

The error indication on pins FCCU_ERR0 (PA[9]) and FCCU_ERR1 (PA[10]) is controlled by GPIO port and FCCU. The mapping of the FCCU output signals on the relevant pins and the associated electrical characteristics are enabled by the GPIO port registers (refer to *Chapter 20: General-purpose I/Os (GPIO)* for signal mapping details). Once the FCCU is mapped onto the pin, the logic level is driven by the FCCU.

The input function FCCU_EIN0 on PA[9] is configured at startup by default. The other FCCU pins are floating at startup (see the IO_Archi for the default configuration of these pins).

Since it is possible to reset FCCU and GPIO port independently, the behavior of FCCU pins is the result of the combined state of GPIO port registers and the FCCU. (Refer to section “Reset interface” in chapter “Fault collection and control unit (FCCU)” for FCCU reset details, and chapter “Reset and Boot” for the state of the FCCU relative to the system during different reset phases).

During functional reset, the application only considers the PAD PA[9] as FCCU error pin. The other FCCU pins are not enabled during functional reset.

The FCCU_STAT[ESTAT] error status flag can be read to show whether the FCCU is in an error state. This flag is written by software to either a 1 (fault) or 0 (operational) when the FCCU is in operational state.

FCCU_EINOUT(EINO): the value of this register depends on the state of the pad. If the input buffer is enabled, the register bit reflects the physical status of the pin as seen by the input buffer. In case the protocol selected for EOUT interface is a toggling one (Dual rail or Time switching), this may not be true due to synchronization in the safety clock domain.

The register reflects the physical status of the pin from application point of view (external status).

FCCU_STAT(PhysicalErrorPin): this register field indicates the actual status of the Error pin as driven by FCCU to the pad based on different FCCU configurations and state of the failure input channels. The value reported in the FCCU_STAT (PhysicalErrorPin) can be different from what is seen at the pad.

5.8.4.1.2 Failure handling

The FCCU is an autonomous module that is responsible for reacting to failure indicators. A different reaction can be configured for each failure source. Overall failure reaction time requires time for detecting, processing, and indicating the error. During this time, SR5E1x can provide wrong results to the system.

Failure sources include:

- All failure indication signals from modules within the MCU.
- Control logic and signals monitored by the FCCU itself (refer to chapter “Fault collection and control unit (FCCU)”).
- Software-initiated failure indications. For example, software signals the FCCU that it has evidence of a failure. Keep in mind that software can also directly influence the state of the FCCU pins.
- External failure input.

Available failure reactions are:

- Assertion of an interrupt (maskable or nonmaskable)
- Resetting the MCU
- Changing the state of the failure indication pins
- Disabling the transmission capabilities of communication controllers (CAN). Note that it is only possible with changing the state of the failure indication pins
- No reaction

Software can read the failure source that caused a fault, either before or after a functional reset (the condition indicators are not volatile). Software can also reset the failure, but the external failure indication stays in failure mode for a configurable minimum time. If necessary, software can also reset the MCU.

5.8.4.2 FCCU registers reset values

The following table shows the reset values of selected FCCU registers, where the reset value is not fully defined in the Fault collection and control unit (FCCU) chapter.

Table 74. Reset values of selected FCCU registers

Register	Reset value
FCCU_CTRL	0x000000C0
FCCU_RF_CFG0	0xFFEBFFFF
FCCU_RF_CFG1	0xFEFFF7FF
FCCU_RF_CFG2	0x0003FFFF
FCCU_RF_CFG3	0x00000000
FCCU_RFS_CFG0	0x00020800
FCCU_RFS_CFG1	0x00000000
FCCU_RFS_CFG2	0x00000000
FCCU_RFS_CFG3	0x00000000
FCCU_RFS_CFG4	0x00000000
FCCU_RFS_CFG5	0x00000000

Table 74. Reset values of selected FCCU registers (continued)

Register	Reset value
FCCU_RFS_CFG6	0x00000000
FCCU_RFS_CFG7	0x00000000
FCCU_RF_S0..S2	0x00000000
FCCU_RF_E0	0x00000120
FCCU_RF_E1..E2	0x00000000
FCCU_RF_TOE0 .. TOE2	0x00000000
FCCU_EINOUT	0x00000030(1)

1. EIN pads are pulled up in testbench.

5.8.4.2.1 FOSU parameters

The FOSU_COUNT (FOSU timeout response time) is 1000 safe clock cycles.

5.8.4.2.2 FCCU failure inputs from other modules

The following table shows the different failure input signals of the FCCU and how they can be tested. Refer to *Table 74* for the default configuration after reset.

Table 75. FCCU failure inputs

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
0	—	—	TEMP_ERROR	Temperature detector	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
1	—	—	LVD_ERROR	Voltage out of range from LVDs (non-destructive reset)	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
2	—	—	HVD_ERROR	Voltage out of range from LVDs (non-destructive reset)	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
3	—	—	DPMC_DCF_SAFETY_ERR	Digital PMC DCF Safety Error	FCCU Fake Fault Injection.	—	Persistent Till Error Negation. The recommended clear mechanism is device reset.
4	—	—	DPMC_VD_BIST	Digital PMC Voltage detector BIST	Error Injection done inside the PMC.	—	Pulse signal. No SW intervention needed to clear it.

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
5	—	—	FLASH_FATAL_ERR	Flash memory initialization error	FCCU Fake Fault Injection.	—	Persistent Till Error Negation. The recommended clear mechanism is device reset.
6	—	—	Flash Reset Error	The pin/flag signals the following unrecoverable errors: – ECC errors on flash internal reads during configuration loading (startup) – ECC errors on flash internal reads during fw copy (startup) – Double ECC errors on KRAM during internal self-check routine (always running)	FCCU Fake Fault Injection.	—	Persistent Till Error Negation. The recommended clear mechanism is device reset.
7	—	—	FLASH_REF_ERR	FLASH read reference error	FCCU Fake Fault Injection.	—	Persistent Till Error Negation. The recommended clear mechanism is device reset.
8	—	—	IWDG1_ResetReq	Independent WDG1 Reset Request	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
9	—	—	IWDG2_ResetReq	Independent WDG2 Reset Request	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
10	—	—	WWDG1_ResetReq	Window watchdog1 Reset Request	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
11	—	—	WWDG2_ResetReq	Window watchdog2 Reset Request	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
12	—	—	FM_PLL_0	PLL0 Loss of Lock (Interrupt)	Error injection mechanism available or programming the PLL dividers inside RCC.	—	Clear status in PLLDIG
13	—	—	FM_PLL_1	PLL1 Loss of Lock (Interrupt)	Error injection mechanism available or programming the PLL dividers inside RCC.	—	Clear status in PLLDIG
14	—	—	CMU_0_OSC	XOSC less than IRC	Error injection mechanism available	—	Clear status in CMU
15	—	—	CMU_0_PLL0	PLL0 out of frequency	Error injection mechanism available	—	Clear status in CMU
16	—	—	CMU_Platform	Sysclk frequency out of range (including HRTIM clock)	Error injection mechanism available	—	Clear status in CMU
17	—	—	CMU_other	Monitoring other internal clocks	Error injection mechanism available	—	Clear status in CMU
18	—	—	—	—	—	—	—
19	—	—	STCU_RF	Bist result - wrong signature (STCU Recoverable Fault)	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
20	—	—	—	—	—	—	—
21	—	—	SYS_RAM_TRIG_0	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
22	—	—	SYS_RAM_TRIG_1	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
23	—	—	SYS_RAM_TRIG_2	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
24	—	—	SYS_RAM_TRIG_3	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
25	—	—	PERIPH_RAM_TRIG_0	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
26	—	—	PERIPH_RAM_TRIG_1	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
27	—	—	PERIPH_RAM_TRIG_2	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
28	—	—	PERIPH_RAM_TRIG_3	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
29	—	—	NVM_TRIG_0	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
30	—	—	NVM_TRIG_1	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
31	—	—	NVM_TRIG_2	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.
32	—	—	NVM_TRIG_3	—	Error injection done inside the MEMU	—	Pulse signal. No SW intervention needed to clear it.

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
33	CEM_10	0	MEMU2_SYS_RAM_SB_OVF	Sys RAM Single Bit Error Table overflow	Error injection done inside the MEMU	—	Clear status in MEMU. SW can write '0' on the Valid bit of any entry of the table, and the MEMU full indication is cleared.
		1	MEMU2_SYS_RAM_UC_OVF	Sys RAM Uncorrectable Error Table overflow			
		2	MEMU2_PERIPH_RAM_SB_OVF	Periph RAM Single Bit Error Table overflow			
		3	MEMU2_PERIPH_RAM_UC_OVF	Periph Uncorrectable Error Table overflow			
		4	MEMU2_NVM_SB_OVF	NVM Single Bit Error Table overflow			
		5	MEMU2_NVM_UC_OVF	NVM Uncorrectable Error Table overflow			
		6	MEMU2_NVM_DBL_OVF	NVM Double correctable Table overflow			

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
34	CEM_11	0	SYS_RAM_OVRFLOW_FIF0_0	—	Error Injection done inside the CEM.	—	Persistent till error negation.
		1	SYS_RAM_OVRFLOW_FIF0_1				
		2	SYS_RAM_OVRFLOW_FIF0_2				
		3	SYS_RAM_OVRFLOW_FIF0_3				
		4	SYS_RAM_OVRFLOW_FIF0_4				
		5	SYS_RAM_OVRFLOW_FIF0_5				
		6	SYS_RAM_OVRFLOW_FIF0_6				
		7	SYS_RAM_OVRFLOW_FIF0_7				
		8	SYS_RAM_OVRFLOW_FIF0_8				
		9	SYS_RAM_OVRFLOW_FIF0_32				
		10	SYS_RAM_OVRFLOW_FIF0_33				
		11	SYS_RAM_OVRFLOW_FIF0_34				
		12	SYS_RAM_OVRFLOW_FIF0_35				
		13	SYS_RAM_OVRFLOW_FIF0_36				
		14	SYS_RAM_OVRFLOW_FIF0_37				
		15	SYS_RAM_OVRFLOW_FIF0_38				
		16	SYS_RAM_OVRFLOW_FIF0_39				
		17	SYS_RAM_OVRFLOW_FIF0_40				
		18	SYS_RAM_OVRFLOW_FIF0_41				
		19	SYS_RAM_OVRFLOW_FIF0_42				
		20	SYS_RAM_OVRFLOW_FIF0_43				
		21	SYS_RAM_OVRFLOW_FIF0_44				

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
35	CEM_12	0	PERIPH_RAM_OVRFLOW_FIF0_0	—	Error Injection done inside the CEM.	—	Persistent till error negation.
		1	PERIPH_RAM_OVRFLOW_FIF0_1				
		2	PERIPH_RAM_OVRFLOW_FIF0_2				
		3	PERIPH_RAM_OVRFLOW_FIF0_32				
		4	PERIPH_RAM_OVRFLOW_FIF0_33				
		5	PERIPH_RAM_OVRFLOW_FIF0_34				
		6	PERIPH_RAM_OVRFLOW_FIF0_35				
		7	PERIPH_RAM_OVRFLOW_FIF0_36				
		8	PERIPH_RAM_OVRFLOW_FIF0_37				
36	—	—	FLASH_OVRFLOW_FIFO_0		FCCU Fake Fault Injection.	—	Persistent till error negation.
37	CEM_9	0	SSCM_XFER_ERR	SSCM transfer error	Error Injection done inside the CEM.	—	Persistent Till Error Negation. The recommended clear mechanism is device reset.
		1	MEMORY REPAIR DCF SAFETY ERROR	Memory Repair Safety Error			
		2	TDM_DCF_SAFETY_ERR	TDM DCF Safety Error			
		3	RCC_DCF_SAFETY_ERR	RCC DCFs + Security Miscellaneous DCF			
38	—	—	ERRIN1	Error from unidirectional input error signal (external failure to MCU)	FCCU Fake Fault Injection.	—	Persistent till error negation from HW
39	—	—	IMA SoC	IMA SoC Active	FCCU Fake Fault Injection.	—	Persistent Till Error Negation

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
40	—	—	SAFE_SWITCH_TO_HSI	Transition to RCOSC in case of critical faults on clock sources	Error injection mechanism available	—	Clear status in RCC.
41	CEM_13	0	SPURIOUS_DEBUG_ACTIVATION	Unexpected activation of JTAG or debug signals	Error injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	SPURIOUS_SSCM_ACTIVATION	Unexpected activation of SSCM CS to DCF clients during run time		—	
		2	SPURIOUS STCU3 ACTIVATION	Unexpected activation of STCU3 during run time		—	
42	—	—	SPURIOUS DFT signals ACTIVATION	Test circuitry Group spurious activation	FCCU Fake Fault Injection.	—	Persistent Till Error Negation.
43	—	—	—	—	—	—	—
44	—	—	Compensation Disable	Indication of undesired deactivation of pad compensation. Fault can affect a distorted transmissions on I/Os associated.(1)	FCCU Fake Fault Injection.	—	Persistent till error negation from HW.
45	—	—	ERRIN0	Error from bidirectional input error signal (external or internal failure to MCU)	FCCU Fake Fault Injection.	—	Persistent till error negation from HW.
46	—	—	Core lock/split change state alarm	Edge detector after dcf loading	FCCU Fake Fault Injection.	—	Persistent Till Error Negation. No SW intervention needed to clear it.

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
47	—	—	Dma lock/split change state alarm	Edge detector after dcf loading	FCCU Fake Fault Injection.	—	Persistent Till Error Negation. No SW intervention needed to clear it.
48	—	—	OTA change state alarm	Edge detector after dcf loading	2 errors: – Boot: no injection available, – NVMC1/2: Fault Injection inside NVMC through internal registers.	—	Persistent Till Error Negation. No SW intervention needed to clear it.
49	—	—	SMPU region	MPU region violation	Error Injection done inside the SMPU.	—	Pulse signal. No SW intervention needed to clear it.
50	—	—	SMPU protocol	The SMPU monitors that the SMPU logic does not alter any signal by comparing in versus out signals.	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
51	—	—	EDC_ECC_Code	EDC after ECC for code NVMC1.	Error injection done inside the NVMC1 through internal registers.	—	Pulse signal. Clear status in NVMC1.
52	—	—	EDC_ECC_Data	EDC after ECC for data NVMC1.	Error injection done inside the NVMC1 through internal registers.	—	Pulse signal. Clear status in NVMC1.
53	—	—	ENC_ERR_FLASH	Flash Encoding Error – Indication of hardware fault resulting in corrupted flash memory access.	Error injection done inside the NVMC1 through internal registers.	—	Pulse signal. Clear status in NVMC1.

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
54	—	—	ADDR_FDBK_ERR_FLASH_CTL	PFlashC Address Feedback Error - alarm indicating the flash controller detected a transaction monitor mismatch when compared to the flash safety feedback outputs.	Error injection done inside the NVMC1 through internal registers.	—	Pulse signal. Clear status in NVMC1.
55	—	—	EDC_ECC_Code	EDC after ECC for Code NVMC2	Error injection done inside the NVMC2 through internal registers.	—	Pulse signal. Clear status in NVMC2.
56	—	—	—	—	—	—	—
57	—	—	ENC_ERR_FLASH	Flash Encoding Error – Indication of hardware fault resulting in corrupted flash memory access.	Error injection done inside the NVMC2 through internal registers.	—	Pulse signal. Clear status in NVMC2.
58	—	—	ADDR_FDBK_ERR_FLASH_CTL	PFlashC Address Feedback Error - alarm indicating the flash controller detected a transaction monitor mismatch when compared to the flash safety feedback outputs	Error injection done inside the NVMC2 through internal registers.	—	Pulse signal. Clear status in NVMC2.

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
59	—	—	e2eECC NVMPC1 Protocol Error	Can be implemented in flash ctl OR of the protocol error on the 2 ports of NVMC1	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
60	—	—	e2eECC NVMPC2 Protocol Error	Can be implemented in flash ctl OR of the protocol error on the 2 ports of NVMC2	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
61	—	—	SRAMC1 EDC after ECC Error	EDC after ECC FCCU Alarm	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
62	—	—	FCCU RAM alarm	FCCU RAM alarm	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
63	—	—	Address/Control EDC/Parity check FCCU Alarm	Address/Control EDC/Parity check FCCU Alarm	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
64	—	—	SRAMC2 EDC after ECC Error	EDC after ECC FCCU Alarm	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
65	—	—	FCCU RAM alarm	FCCU RAM alarm	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.
66	—	—	Address/Control EDC/Parity check FCCU Alarm	Address/Control EDC/Parity check FCCU Alarm	FCCU Fake Fault Injection.	—	Pulse signal. No SW intervention needed to clear it.

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
67	CEM_0	0	e2eECC Data Correctable Error Core1 AXIM	—	Error Injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	e2eECC Data Correctable Error Core1 AHBM				
		2	e2eECC Data Correctable Error Core2 AXIM				
		3	e2eECC Data Correctable Error Core2 AHBM				
		4	e2eECC Data Correctable Error HSM AHB				
		5	e2eECC Data Correctable Error DMA1 AHB Memory				
		6	e2eECC Data Correctable Error DMA1 AHB Peripheral				
		7	e2eECC Data Correctable Error DMA2 AHB Memory				
		8	e2eECC Data Correctable Error DMA2 AHB Peripheral				
68	CEM_1	0	e2eECC Data Uncorrectable Error Core1 AXIM	—	Error Injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	e2eECC Data Uncorrectable Error Core1 AHBM				
		2	e2eECC Data Uncorrectable Error Core2 AXIM				
		3	e2eECC Data Uncorrectable Error Core2 AHBM				
		4	e2eECC Data Uncorrectable Error HSM AHB				
		5	e2eECC Data Uncorrectable Error DMA1 AHB Memory				
		6	e2eECC Data Uncorrectable Error DMA1 AHB Peripheral				
		7	e2eECC Data Uncorrectable Error DMA2 AHB Memory				
		8	e2eECC Data Uncorrectable Error DMA2 AHB Peripheral				

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
69	CEM_2	0	e2eECC Protocol Error Core1 AXIM	—	Error Injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	e2eECC Protocol Error Core1 AHBM				
		2	e2eECC Protocol Error Core2 AXIM				
		3	e2eECC Protocol Error Core2 AHBM				
		4	e2eECC Protocol Error HSM AHB				
		5	e2eECC Protocol Error DMA1 AHB Memory				
		6	e2eECC Protocol Error DMA1 AHB Peripheral				
		7	e2eECC Protocol Error DMA2 AHB Memory				
		8	e2eECC Protocol Error DMA2 AHB Peripheral				
70	—	—	AXI watchdog	—	No Injection Mechanism Available.	—	Pulse signal. No SW intervention needed to clear it.
71	CEM_3	0	e2eECC Data Correctable Error Cores AHBP	—	Error Injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	e2eECC Data Correctable Error AHB1				
		2	e2eECC Data Correctable Error AHB2				
		3	e2eECC Data Correctable Error APB1				
		4	e2eECC Data Correctable Error APB2				
		5	e2eECC Data Correctable Error HRTIM1 AXI				
		6	e2eECC Data Correctable Error HRTIM2 AXI				

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
72	CEM_4	0	e2eECC Data Uncorrectable Error Cores AHBP	—	Error Injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	e2eECC Data Uncorrectable Error AHB1				
		2	e2eECC Data Uncorrectable Error AHB2				
		3	e2eECC Data Uncorrectable Error APB1				
		4	e2eECC Data Uncorrectable Error APB2				
		5	e2eECC Data Uncorrectable Error HRTIM1 AXI				
		6	e2eECC Data Uncorrectable Error HRTIM2 AXI				
73	CEM_5	0	e2eECC Protocol Error Cores AHBP	—	Error Injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	e2eECC Protocol Error AHB1				
		2	e2eECC Protocol Error AHB2				
		3	e2eECC Protocol Error APB1				
		4	e2eECC Protocol Error APB2				
		5	e2eECC Protocol Error HRTIM1 AXI				
		6	e2eECC Protocol Error HRTIM2 AXI				
74	CEM_6	0	Protection violation AHB1	—	Error Injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	Protection violation AHB2				
		2	Protection violation APB1				
		3	Protection violation APB2				
75	—	—	RCCU_CORE_alarm	RCCUS for Cores lockstep	FCCU Fake Fault Injection.	—	Persistent till error negation. No SW intervention needed to clear it.
76	—	—	RCCU_DMA_alarm	RCCUS for DMA lockstep	FCCU Fake Fault Injection.	—	Persistent till error negation. No SW intervention needed to clear it.

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
77	CEM_7	0	AHB1_Bridge_Alarm	from RCCUSs for Slaves Dataless Duplication lockstep	Error Injection done inside the CEM.	—	Persistent till error negation. No SW intervention needed to clear it.
		1	AHB2_Bridge_Alarm				
		2	APB1_Bridge_Alarm				
		3	APB2_Bridge_Alarm				
		4	AHBS_Bridge_Alarm				
		5	HRTIM1_Bridge_Alarm				
		6	HRTIM2_Bridge_Alarm				
		7	NVMC1_Bridge_Alarm				
		8	NVMC2_Bridge_Alarm				
		9	RAMC1_Bridge_Alarm				
		10	RAMC2_Bridge_Alarm				
78	—	—	CORE1_LOCKUP	—	FCCU Fake Fault Injection.	—	Persistent till next reset.
79	—	—	CORE2_LOCKUP	—	FCCU Fake Fault Injection.	—	Persistent till next reset.
80	CEM_8	0	I-TCM Core1 Address feedback Err	—	Error Injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	D0-TCM Core1 Address feedback Err				
		2	D1-TCM Core1 Address feedback Err				
		3	I-TCM Core1 EDC after ECC				
		4	D0-TCM Core1 EDC after ECC				
		5	D1-TCM Core1 EDC after ECC				
		6	I-TCM Core2 Address feedback Err				
		7	D0-TCM Core2 Address feedback Err				
		8	D1-TCM Core2 Address feedback Err				
		9	I-TCM Core2 EDC after ECC				
		10	D0-TCM Core2 EDC after ECC				
		11	D1-TCM Core2 EDC after ECC				

Table 75. FCCU failure inputs (continued)

FCCU Channel	CEM instance	CEM reg bit	Failure	Failure Description	Error Injection check	Reference Manual related information	Clear mechanism in the Fault Source
81	CEM_14	0	Upsizer error - Core1 AHB	Fault triggered Decode error response is seen on the bus.	Error Injection done inside the CEM.	—	Pulse signal. No SW intervention needed to clear it.
		1	Upsizer error - Core2 AHB				
		2	Upsizer error - HSM				
		3	Upsizer error - DMA1 AHBP				
		4	Upsizer error - DMA1 AHBM				
		5	Upsizer error - DMA2 AHBP				
		6	Upsizer error - DMA2 AHBM				

1. Compensation cell reduces the spread of some circuit parameters (slew rate of the output signal and the output impedance) in the IO buffers over temperature, process and voltage. Compensation cell can remain disabled until supplies have reached LVD290 threshold.

5.8.5 CEM configuration

The SR5E1x contains 15 CEM instances (CEM0 to CEM14). Each instance has only one group (Group0).

5.8.6 REG_PROT configuration

The AHB and APB bridges include a hardware mechanism to protect module registers from unwanted modifications.

The protection is applied at the level of the memory slots in the AHB and APB bridges. The memory space reserved for each slot is 1 KB, that in most cases corresponds with the memory space reserved for an IP module. Inside each IP slot, registers are protected with 32-bit granularity.

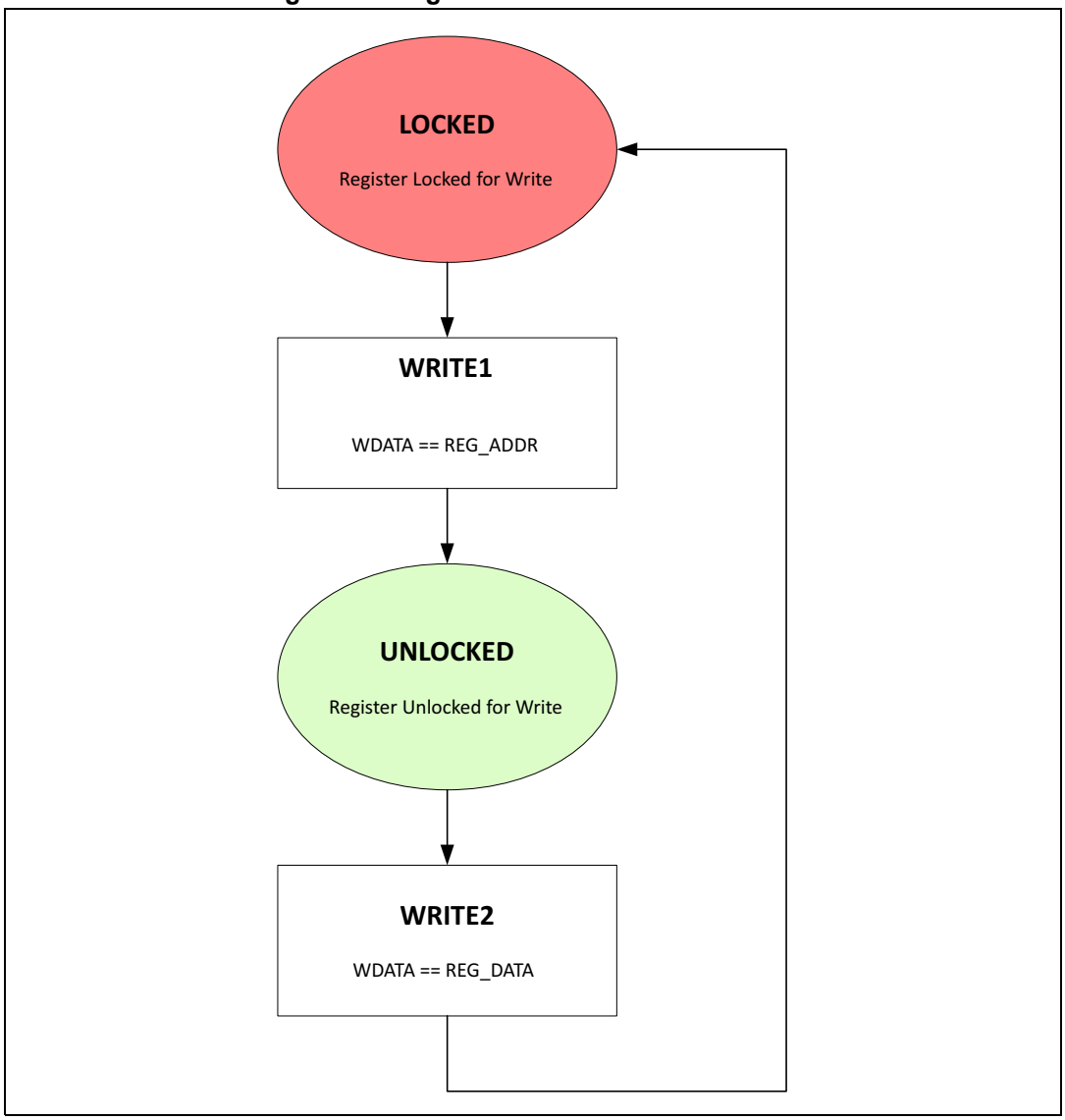
The protection mechanism works like a firewall for write operations, basing on the lock status of the register. For this reason, the Register Protection feature is also referred to as Register Hardware Lock.

To write in a protected register, the software has to first unlock it by writing a password equal to the register address.

The modification of a protected register consists of a two-step write operation, as shown in *Figure 13*:

- The first write (WRITE1) has to be performed on the register, providing as data the address of the register. This unlocks the register for write.
- The second write (WRITE2) is the effective write to the register with the desired data.

Figure 13. Register hardware lock for writes



Note: If a register is unlocked by bus master1 and another master (that is master2) tries to write in it, master2 gets an error response.

The protected registers for each IP have been selected at design time. The list of protected register in each module is shown from *Table 76: Protected GPIOx registers* to *Table 85: MBIST partitions*.

The register protection mechanism is not enabled by default. An enable bit for each AHB and APB bridge is present in REG_HWLOCK_DCF (see *REG_HWLOCK_DCF*).

5.8.6.1 GPIOx protected registers

The following table lists the GPIOx, x=A..I, registers that can be protected.

Table 76. Protected GPIOx registers

Register	Register size (bits)	Offset from module base address
MODER	—	0x00000000
OTYPER	—	0x00000004
OSPEEDR	—	0x00000008
PUPDR	—	0x0000000C
AFRL	—	0x00000020
AFRH	—	0x00000024

5.8.6.2 RCC protected registers

The following table lists the Reset & Clock Control (RCC) protected registers.

Table 77. Protected RCC registers

Register	Register size (bits)	Offset from module base address
RCC_CR	—	0x00000000
RCC_ICSR	—	0x00000004
RCC_CFGR	—	0x00000008
RCC_PLLCFGR	—	0x0000000C
RCC_CIER	—	0x00000018
RCC_CIFR	—	0x0000001C
RCC_CICR	—	0x00000020
RCC_CCIPR1	—	0x00000024
RCC_CCIPR2	—	0x00000028
RCC_LSCFGR	—	0x0000002C
RCC_CRRCR	—	0x00000030
RCC_AHB1LRSTR	—	0x00000050
RCC_AHB1HRSTR	—	0x00000054
RCC_AHB2LRSTR	—	0x00000058
RCC_AHB2HRSTR	—	0x0000005C
RCC_APB1LRSTR	—	0x00000060
RCC_APB2LRSTR	—	0x00000068
RCC_APB2HRSTR	—	0x0000006C
RCC_AHB1LENR	—	0x00000070
RCC_AHB1HENR	—	0x00000074
RCC_AHB2LENR	—	0x00000078
RCC_AHB2HENR	—	0x0000007C

Table 77. Protected RCC registers (continued)

Register	Register size (bits)	Offset from module base address
RCC_APB1LENR	—	0x00000080
RCC_APB2LENR	—	0x00000088
RCC_APB2HENR	—	0x0000008C
RCC_AHB1LSMENR	—	0x00000090
RCC_AHB1HSMENR	—	0x00000094
RCC_AHB2LSMENR	—	0x00000098
RCC_AHB2HSMENR	—	0x0000009C
RCC_APB1LSMENR	—	0x000000A0
RCC_APB2LSMENR	—	0x000000A8
RCC_APB2HSMENR	—	0x000000AC

The following table lists the Clock Monitoring Units (CMU) protected registers.

Table 78. Protected CMUx, x= 0..1, registers

Register	Register size (bits)	Offset from module base address
CSR	—	0x00000000
HFREFR	—	0x00000008
LFREFR	—	0x0000000C
ISR	—	0x00000010

The following table lists the Power Management Controller (PMC) protected registers.

Table 79. Protected PMC registers

Register	Register size (bits)	Offset from module base address
REE_LV0	—	0x00000004
RES_LV0	—	0x00000008
FEE_LV0	—	0x00000010
REE_LV1	—	0x00000024
RES_LV1	—	0x00000028
FEE_LV1	—	0x00000030
REE_HV0	—	0x00000044
RES_HV0	—	0x00000048
FEE_HV0	—	0x00000050
VSIO	—	0x00000104

Table 79. Protected PMC registers (continued)

Register	Register size (bits)	Offset from module base address
MISC_CTRL_REG	—	0x00000248
REE_TD	—	0x00000304
RES_TD	—	0x00000308
CTL_TD	—	0x0000030C
FEE_TD	—	0x00000318
BIST_CTRL	—	0x000003D4

The following table lists the (MEMU) protected registers.

Table 80. Protected MEMU registers

Register	Register size (bits)	Offset from module base address
MEMU_CTRL	—	0x00000000

The following table lists the Indirect Memory Adressing (IMA) protected registers.

Table 81. Protected IMA registers

Register	Register size (bits)	Offset from module base address
IMA_ENABLE	—	0x00000004

The following table lists the System Control (SYSCTRL) protected registers.

Table 82. Protected SYSCTRL registers

Register	Register size (bits)	Offset from module base address
EXTICR_3_0	—	0x00000008
EXTICR_7_4	—	0x0000000C
EXTICR_11_8	—	0x00000010
EXTICR_15_12	—	0x00000014

The following table lists the PLL protected registers.

Table 83. Protected PLL registers

Register	Register size (bits)	Offset from module base address
PLL0CR	—	0x00000000
PLL0SR	—	0x00000004

Table 83. Protected PLL registers (continued)

Register	Register size (bits)	Offset from module base address
PLL0DV	—	0x00000008
PLL1CR	—	0x00000020
PLL1SR	—	0x00000024
PLL1DV	—	0x00000028
PLL1FM	—	0x0000002C
PLL1FD	—	0x00000030

5.8.7 STCU configuration

The following table lists the CBIST collectors (Built-in Self-Test on RCCUs monitoring duplicated logic).

Table 84. CBIST collectors

CBIST DCF	Comparators protected by CBIST	CBIST ID
STCU_CBIST0_PTR	Core1, Core2, ITCMs, DTCMs	0
STCU_CBIST1_PTR	DMA1, DMA2	1
STCU_CBIST2_PTR	AHB1, AHB2	2
STCU_CBIST3_PTR	APB1	3
STCU_CBIST4_PTR	APB2	4
STCU_CBIST5_PTR	HRTimer_1	5
STCU_CBIST6_PTR	HRTimer_2	6
STCU_CBIST7_PTR	Flash controllers: NVMC1, NVMC2	7
STCU_CBIST8_PTR	RAM controllers: RAMC1, RAMC2	8
STCU_CBIST9_PTR	MEMU	9

The following table lists the MBIST partitions (Built-in Self-Test on memory cuts).

Table 85. MBIST partitions

MBIST number	Location	Partition	Addressable	Start address in MEMU	End address in MEMU
0	Core1	ICache_1	No	0x6000_0000	0x6000_FFFF
1		ICache_0	No		
2		ICache_Tag_1	No	0x6001_0000	0x6001_FFFF
3		ICache_Tag_0	No		
4		DCache_0_3	No	0x6002_0000	0x6002_FFFF
5		DCache_0_2	No		
6		DCache_0_1	No		
7		DCache_0_0	No		
8		DCache_1_3	No	0x6003_0000	0x6003_FFFF
9		DCache_1_2	No		
10		DCache_1_1	No		
11		DCache_1_0	No		
12		DCache_Tag3	No	0x6004_0000	0x6004_FFFF
13		DCache_Tag2	No		
14		DCache_Tag1	No		
15		DCache_Tag0	No		
16	Core2	ICache_1	No	0x6005_0000	0x6005_FFFF
17		ICache_0	No		
18		ICache_Tag_1	No	0x6006_0000	0x6006_FFFF
19		ICache_Tag_0	No		
20		DCache_0_3	No	0x6007_0000	0x6007_FFFF
21		DCache_0_2	No		
22		DCache_0_1	No		
23		DCache_0_0	No		
24		DCache_1_3	No	0x6008_0000	0x6008_FFFF
25		DCache_1_2	No		
26		DCache_1_1	No		
27		DCache_1_0	No		
28		DCache_Tag3	No	0x6009_0000	0x6009_FFFF
29		DCache_Tag2	No		
30		DCache_Tag1	No		
31		DCache_Tag0	No		
32	CAN SUB0	CAN Subsystem 0 RAM	Yes	0x4200_C000	0x4200_FFFF

Table 85. MBIST partitions (continued)

MBIST number	Location	Partition	Addressable	Start address in MEMU	End address in MEMU
33	Flash	RAM	No	0x600C_0000	0x600C_27FF
34	Flash	ROM	—	—	—
35	ETF	ETF RAM	No	0x600E_0000	0x600E_FFFF
36	HSM	PRAM_0	Yes	0x0100_0000	0x0100_1FFF
37		PRAM_2	Yes	0x0100_2000	0x0100_9FFF
38		PRAM_1	Yes		
39		SBI_Cache_1	No	0x600A_0000	0x600A_0FFF
40		SBI_Cache_0	No		
41		SBI_Tag_1	No	0x600B_0000	0x600B_07FF
42		SBI_Tag_0	No		
43		MPAES	No	0x600D_0000	0x600D_FFFF
44	—	System RAM_1_1	Yes	0x2401_0000	0x2401_FFFF
45	—	System RAM_1_0	Yes	0x2400_0000	0x2400_FFFF
46	—	System RAM_2_1	Yes	0x2403_0000	0x2403_FFFF
47	—	System RAM_2_0	Yes	0x2402_0000	0x2402_FFFF
48	Core1	I_TCM	Yes	0x5A00_0000	0x5A00_7FFF
49		D0_TCM_1	Yes	0x5C00_4000	0x5C00_7FFF
50		D0_TCM_0	Yes	0x5C00_0000	0x5C00_3FFF
51		D1_TCM_1	Yes	0x5C00_C000	0x5C00_FFFF
52		D1_TCM_0	Yes	0x5C00_8000	0x5C00_BFFF
53	Core2	I_TCM (NO_LS)	Yes	0x5A04_0000	0x5A04_7FFF
		I_TCM (LS)	Yes	0x5A00_8000	0x5A00_FFFF
54		D0_TCM_1 (NO_LS)	Yes	0x5C04_4000	0x5C04_7FFF
		D0_TCM_1 (LS)	Yes	0x5C01_4000	0x5C01_7FFF
55		D0_TCM_0 (NO_LS)	Yes	0x5C04_0000	0x5C04_3FFF
		D0_TCM_0 (LS)	Yes	0x5C01_0000	0x5C01_3FFF
56		D1_TCM_1 (NO_LS)	Yes	0x5C04_C000	0x5C04_FFFF
		D1_TCM_1 (LS)	Yes	0x5C01_C000	0x5C01_FFFF
57		D1_TCM_0 (NO_LS)	Yes	0x5C04_8000	0x5C04_BFFF
		D1_TCM_0 (LS)	Yes	0x5C01_8000	0x5C01_BFFF

5.8.8 ECC

Error correcting codes are used for end-to-end protection from cores to system storage as well as for individual protection of peripheral RAMs.

5.8.8.1 ECC for storage

All storage used in normal operation is protected by ECC with:

- SEC/DED (single bit error correction and double bit error detection) for core, local memory, peripheral RAM.
- SEC/DED (single bit error correction, double bit error detection) for CODE Flash memory.
- DEC/TED (double bit error correction, triple bit error detection) for DATA Flash memory.

The following exceptions exist:

- SRAM is protected by SEC/DEC ECC in PRAM controller only during partial writes (RMW operations) otherwise it is protected by E2E-ECC mechanism.
- The HSM's AES program extension RAM are not protected by ECC.

A list showing the implementation of RAMs with ECC (including address protection) and for SR5E1x is shown in the following table.

Table 86. ECC RAM implementations

Location	Memory	MEMU classification	ECC	Address is ECC	Muxing Factor	Reacts to uncorrectable error
Cortex®-M7 Core 1	I-Cache	System RAM	SEC/DED	No	4	Y
	ITAG	System RAM	SEC/DED	No	4	Y
	D-Cache	System RAM	SEC/DED	No	4	Y
	DTAG	System RAM	SEC/DED	No	4	Y
	ITCM	System RAM	SEC/DED	No	8	Y
	DTCM	System RAM	SEC/DED	No	8	Y
Cortex®-M7 Core 2	I-Cache	System RAM	SEC/DED	No	4	Y
	ITAG	System RAM	SEC/DED	No	4	Y
	D-Cache	System RAM	SEC/DED	No	4	Y
	DTAG	System RAM	SEC/DED	No	4	Y
	ITCM	System RAM	SEC/DED	No	8	Y
	DTCM	System RAM	SEC/DED	No	8	Y
CAN	CAN	Peripheral	SEC/DED	No	8	Y
HSM	PRAM	Peripheral	SEC/DED	No	8	Y
	I-Cache	Peripheral	SEC/DED	No	4	Y
	ITAG	Peripheral	SEC/DED	No	4	Y
	C3	Peripheral	—	No	4	N
Platform	SRAM	System RAM	E2E_ECC(1)	No	16	Y

1. SEC/DEC for partial write operation (RMW).

As shown in *Table 86: ECC RAM implementations*, some memories, particularly system storage, use an ECC computed over data to detect faults (no/wrong/multiple selection). In addition, these same memories include dedicated measures against addressing and control faults (such as address/control feedback). This is different for storage related to PeMos. PeMos, in general, do not include additional measures to detect them. All ECC calculations for RAMs in PeMos are accomplished by logic outside of the RAM, not in the RAM block.

6 Reset and boot

6.1 Introduction

This chapter describes the reset and boot phase, from the time a power on or reset is applied, till the completion of the Boot Assist Flash (BAF) code.

After reset, Core_1 is woken-up, Core_2 and HSM may be woken up (refer to details in *Chapter 7: Device configuration format (DCF) records*).

The following memory elements are involved in the boot-up process:

- UTEST flash memory
- TEST flash memory
- Boot Assist Flash (BAF)

6.1.1 TEST flash memory block

The TEST flash memory block contains Device Configuration Format (DCF) records used to hold trim values and other variables, as well as general device configuration information.

The trim values are for:

- Adjusting low-voltage and high-voltage detect circuit trip points
- Temperature sensor adjustments
- Power-on reset voltage trip point
- Analog-to-digital adjustments
- Internal RC oscillator (IRCOSC) trim values
- Power management configuration for the devices supporting different regulation schemes

Note: The DCF records are written by the device manufacturer and programmed into TEST flash memory during production testing. Further programming of the TEST flash is disabled at the end of the factory test cycle.

6.1.2 UTEST flash memory block

The UTEST flash memory block also contains DCF records. Some UTEST DCF records are written by the factory and programmed during production testing. Others are written by the end user and programmed at the same time application code is programmed into the flash memory. UTEST DCF records are used to set up various control and configuration registers, including the Self-test control unit (STCU3) and default memory configuration.

6.1.3 Boot assist flash

The Boot Assist Flash (BAF) contains factory code to facilitate the device boot-up procedure.

Refer to *Chapter 18: Boot assist flash (BAF)* for further details on this procedure.

6.2 Modules used in reset sequence

The modules involved in the SR5E1x reset sequence are:

- Power management controller (PMC)
- Reset and clock control module (RCC)
- System status and configuration module (SSCM)
- Self-test control unit (STCU3)

6.2.1 Power management controller

The Power management controller (PMC) controls and monitors:

- Its own supply voltage
- The supply voltages to all the high- and low-voltage detect circuits
- The trip points for all the high- and low-voltage detect circuits
- The power supplies and reference voltages to the analog-to-digital converter
- The major power supplies to the SR5E1x

6.2.2 Reset and clock control module (RCC)

The reset and clock control module (RCC) is a complex state machine that begins sequencing the SR5E1x through the initial steps of the reset process. The RCC does not execute program code. It is a state machine that centralizes the different reset sources and manages the reset sequence. Reset sources are organized into two categories: destructive and functional.

The RCC is responsible for configuring the execution mode, including the delivery of reset vectors to all the SR5E1x cores.

Refer to *Figure 14: Reset sequence* for more information on the RCC reset sequence.

6.2.2.1 Destructive resets

A destructive reset source is related to a critical error or dysfunction, usually caused by a hardware malfunction. When a destructive reset event occurs, software recovery is not possible and the contents of memory are assumed to be unknown. A full device reset sequence starting from PHASE0 is therefore applied to the device, ensuring a safe startup state for both digital and analog modules.

Examples of destructive resets are:

- Power-on reset
- External reset
- Low voltage detection
- Reset escalation (when too many functional resets occur simultaneously)

6.2.2.2 Functional resets

A functional reset source is related to a less critical error or dysfunction usually not associated with hardware malfunction. When a functional reset event occurs, a partial reset sequence is applied to the device starting with PHASE1[FUNC]: most digital modules are reset normally and the state of analog modules, specific digital modules (for example, debug and flash memory modules) and system memory content is preserved.

Examples of functional resets are:

- Software reset from RCC

- Boundary scan instructions
- Alarm thresholds from low voltage and temperature detection

6.2.3 System status and configuration module

During the reset phase, the SSCM reads the DCF records in the TEST and UTEST flash memory areas to distribute within the device the preassigned configuration.

The DCF records are primarily for setting up the memory, configuring the self-test control unit (STCU3), and providing initial device configuration values.

The boot vectors for the cores are loaded inside RCC as DCF records by the SSCM. The start addresses are written into the RCC registers, which, in turn, pass them to the proper CPU at the end of the reset phase.

6.2.4 Self-test control unit

The self-test control unit (STCU3) is a self-contained module that runs a specific logic built-in self-test (CBIST) and a memory built-in self-test (MBIST). The DCF record can be used to select which individual tests are run for CBIST and MBIST and disable tests for modules and memory elements that are not going to be used in user applications, thus shortening device boot-up time.

6.3 Reset sequence

The power-up reset sequence always begins with the application of power and follows different sequences depending on the condition of the SR5E1x device and whether various modes are enabled from settings in the DCF records. For instance, the SR5E1x device enters Serial Boot mode (the Serial Boot Loader receives startup code and begins program execution) if there is not a valid address in C1_APPLI_ADDR DCF.

6.3.1 Power-on and reset generation module

When power is applied to the SR5E1x, the Reset and Clock Control Module (RCC) advances the device through a series of steps shown in *Figure 14: Reset sequence*.

Reset and clock control module (RCC) starts the system status and control module (SSCM), which continues the boot-up process after the RCC enters the IDLE state.

The STCU3 is reset and offline self-test (if enabled) is triggered on any power-on, or destructive reset (including external reset through RESETN pad).

Figure 14. Reset sequence

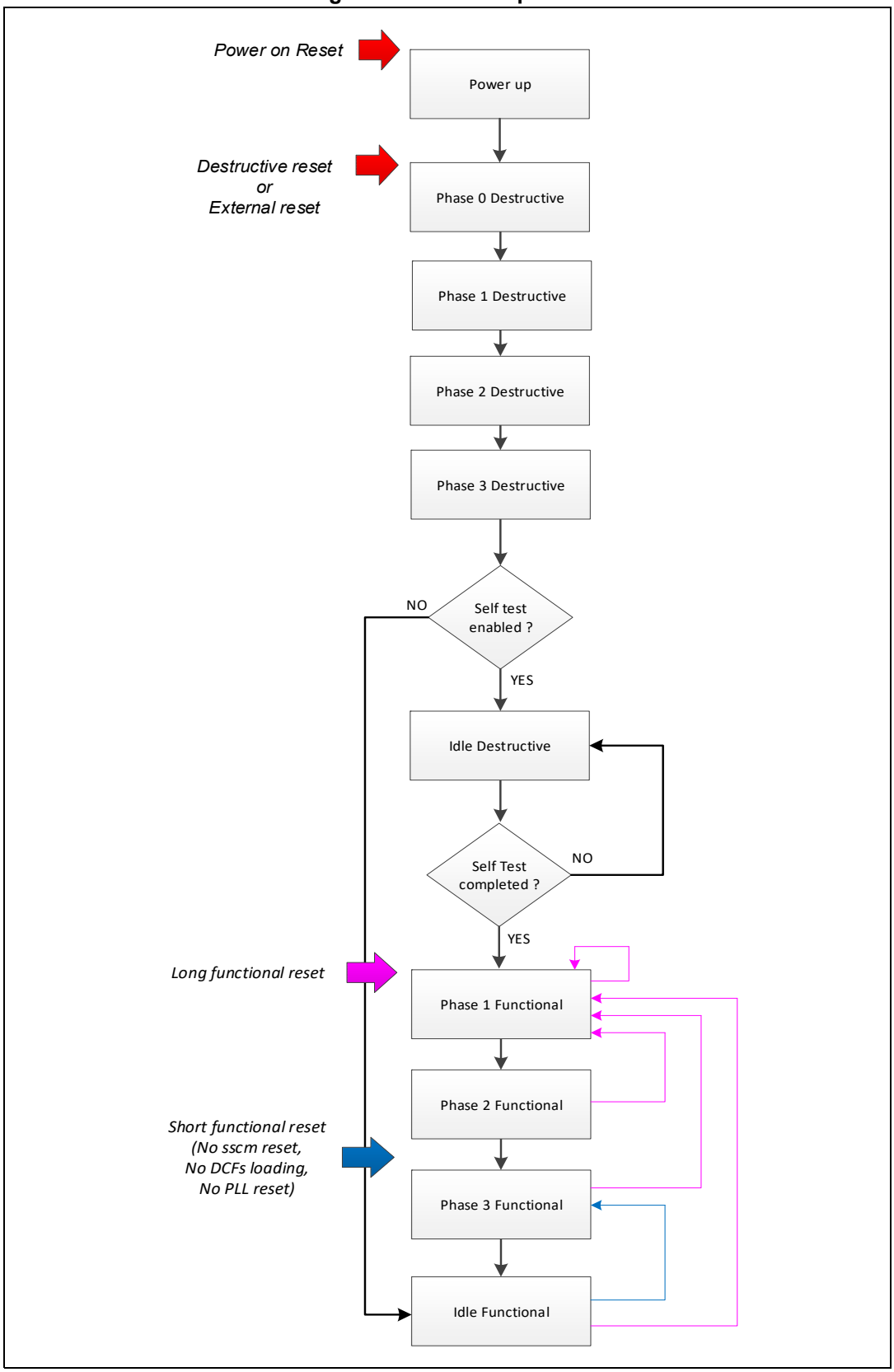


Table 87. Module status during reset phases

Module	Phase									
	Power Up	Destructive					Functional			
		P0	P1	P2	P3	IDLE(1)	P1	P2	P3	IDLE
Internal RCOSC	RESET	ON	ON	ON	ON	ON	ON	ON	ON	ON
PMC	ON(2)	ON	ON	ON	ON	ON	ON	ON	ON	ON
RCC (Reset State Machine)	RESET	ON(3)	ON	ON	ON	ON	ON	ON	ON	ON
Temperature Thresholds	RESET	ON	ON	ON	ON	ON	ON	ON	ON	ON
NVM	RESET	RESET	Running NVM reset phase	ON	ON	BIST	RESET	ON	ON	ON
XOSC	RESET	RESET	RESET	RESET	(ON(4))	(ON(4))	(ON(4))	(ON(4))	(ON(4))	(ON(4))
PLL	RESET	RESET	RESET	RESET	RESET	ON(5)	RESET	RESET	RESET OR ON(6)	RESET OR ON(6)
Device Configuration (for example DCF)	RESET	RESET	HOLD Reset Value	HOLD Reset Value	Loaded from NVM	HOLD Loaded Value	HOLD Loaded Value	HOLD Loaded Value	HOLD Loaded Value	HOLD Loaded Value
SSCM	RESET	RESET	RESET	RESET	ON	ON	RESET	RESET	ON	ON
FCCU	RESET	RESET	RESET	ON	ON	ON	ON	ON	ON	ON
CEMs	RESET	RESET	RESET	ON	ON	ON	ON	ON	ON	ON
MEMU2	RESET	RESET	RESET	ON	ON	ON	ON	ON	ON	ON
STCU	RESET	RESET	RESET	CONFIG	CONFIG	ON	IDLE	IDLE	IDLE	IDLE
Boot CPU	RESET	RESET	RESET	RESET	RESET	RESET	RESET	RESET	RESET	ON
PASS(7)	RESET	ON	ON	ON	ON	ON	ON	ON	ON	ON
IWDGs	RESET	RESET	RESET	RESET	RESET	RESET	RESET	RESET	RESET	ON/ IDLE(8)
Other cores	RESET	RESET	RESET	RESET	RESET	BIST	RESET	RESET	RESET	OFF
Other modules	RESET	RESET	RESET	RESET	RESET	BIST	RESET	RESET	RESET	OFF

1. Self test is run if enabled via DCF, upon a Destructive Reset, including external reset from pad.
2. Set of voltage monitors used to detect when the minimum levels are reached.
3. RCC Reset State Machine is reset only upon a power-on.
4. XOSC activation is set via DCF.
5. PLL0 can be enabled during the offline self-test execution.
6. PLL setting is maintained in case of a Short Functional reset.

7. PASS is reset upon a power-on, as it must be active in order to maintain the debug enabling status across a reset sequence.
8. IWDGs can be activated in HW via DCF.

6.3.2 Power-up phase: power stabilization

When a power-on reset event occurs (initial application of power), the reset and clock control module (RCC) causes the device to enter the power-up phase of the reset sequence (refer to *Figure 14: Reset sequence*). The only method to enter the power-up phase is from a power-on reset event.

The device remains in the power-up phase until the power management controller module (PMC) determines that all power supplies are in their respective operational ranges.

The supplies for the initial configuration are:

- VDD_HV_IO: high voltage supply for I/Os and core module, including power management unit, internal RC oscillator (IRCOSC).
- VDD_LV: low voltage supply for digital module, including low voltage supply for flash module
- VDD_HV_FLA: high voltage supply for flash module.

Peripheral supplies such as VDD_HV_SAR, VDD_HV_SD_DAC_COMP, VDD_HV_ODC do not gate the power-up process. The low- and high-voltage detect circuits for these and other power supplies are managed by the PMC, which provides enable and status bits for these voltage detection circuits.

The PMC sends a signal to the reset and clock control module (RCC) when power stabilization has been achieved. The RCC module then advances to the PHASE0 state (refer to *Figure 14: Reset sequence*). During the power-up phase, the PMC does the following:

- Drives to a logic ‘0’ all VDs (voltage detectors) signals monitored by the power management control (PMC) module.
- Monitors its own power supply voltage (VDD_HV_PMC) to determine that the applied voltage is within a specified range.
- Monitors the core power supply voltage (VDD_LV) to determine that the applied voltage is within a specified range.
- Monitors the low- and high-voltage detect circuit power supply voltage to determine that the applied voltage is within a specified range.
- Holds the outputs of the internal low- and high-voltage detect circuits at a ground state until the PMC determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Enables all low- and high-voltage detect circuits once all power supplies are at their functional levels.
- Begins to monitor the various power supplies using the high- and low-voltage detect circuits.
- Provides status information to the reset and clock control module (RCC) indicating whether or not power is properly applied.

To exit power-up and enter PHASE0:

- All enabled destructive resets must be processed.
- All power supplies must be at their functional voltages.
- The PMC must signal the RCC that all power supplies are at the required levels.

6.3.3 PHASE0 phase: analog supply initial configuration

This phase is entered:

- On exit from power-up phase.
- On a destructive reset event, including RESETN pin falling edge detection except for the initial power-up sequence.
  - As soon as a destructive reset is detected, the reset state machine goes in PHASE0, immediately from PHASE1[DEST], PHASE2[DEST], PHASE3[DEST], IDLE[DEST], PHASE1[FUNC], PHASE2[FUNC], PHASE3[FUNC] or IDLE[FUNC].

During PHASE0:

- All digital modules are reset, including safety, security, and test modules.
- All trimming bits for analog modules are reset.
- Startup of the internal analog modules (IRCOSC, Flash memory, and I/Os) begins:
  - PMC determines that proper voltages are applied.
  - I/O pins are ensured correctly configured

To exit PHASE0 and enter PHASE1[DEST]:

- All enabled destructive resets must be processed.
- Gating PHASE0 must be released.
- All processes initiated in PHASE0 must be completed. This generally includes startup of the internal analog modules: PMC, IRCOSC, and I/Os.

Note: The core voltage must rise above the LVD119 upper threshold to exit PHASE0. This ensures significant hysteresis on low voltage supply avoiding spurious low voltage detection during the power-up process.

6.3.4 PHASE1[DEST] phase: temporization setup

This phase is entered on exit from PHASE0.

During PHASE1[DEST]: temporization to allow the correct NVM reset is applied.

Test modules are available during PHASE1[DEST], but their functionality is limited by the security module.

All PHASE1[DEST] tasks must be completed before entering PHASE2[DEST].

6.3.5 PHASE2[DEST] phase: flash initial configuration

This phase is entered on exit from PHASE1[DEST].

During PHASE2[DEST]:

- Reset is released to the flash memory module by the reset generation module.
- The flash memory module starts its initial configuration process.

- The flash memory initialization is performed by a state machine internal to the flash module.
- A boot timeout inside RCC monitors the flash memory configuration execution so that a reset occurs in the event of configuration failure.

Note: The RCC boot timeout is hardware configured to generate a destructive reset in case of flash memory failure during the flash boot process. In this case, the software can read the cause of the reset by the RCC register flag DESR.D_SWT_Boot.

All PHASE2[DEST] tasks must be completed before entering PHASE3[DEST].

6.3.6 PHASE3[DEST] phase: device configuration

This phase is entered on exit from PHASE2[DEST].

During PHASE3[DEST], the system status and configuration module (SSCM) starts by retrieving the device configuration format (DCF) record from the TEST and UTEST flash memory areas, and uses the configuration and initialization information therein to:

- Configure the self-test control unit (STCU3) and the initial memory map of the device. This information enables tests that the STCU runs and also provides the memory map setup information so that the STCU can test memory.
- Transfer reset vectors for the various cores, programmed as DCF records, to the reset and clock control module (RCC).
- Write trim values for analog modules including the analog-to-digital converter, low- and high-voltage detect circuits, and the temperature sensor to their respective registers.

The DCF record programmed into the TEST flash memory area is written by the manufacturer. The portion of the DCF record programmed into the UTEST flash memory is usually written by the customer, although there is a default DCF record programmed into both the TEST and the UTEST flash memory areas during device production testing.

The DCF record also provides configuration information for the following:

- Trimming of the analog modules (for example, voltage regulator, voltage detectors, I/Os, and oscillator)
  - Trim values are in DCF records in the TEST flash memory area
  - Trim values are determined during production test and programmed into the TEST flash memory area
  - Security configuration
- Application configuration bits:
  - Watchdog configuration
  - Safety execution directives (self-test, CBIST/MBIST)
  - Boot Assist Flash options
  - Oscillator startup during reset sequence configuration

To exit PHASE3[DEST]: All processes that need to be completed in PHASE3[DEST] must be finished.

This phase depends on actual trim values, especially those used for safety execution directives. This device configuration is maintained until the next internal power-on reset. Each time a destructive reset event occurs, the SSCM takes values from the DCF record and writes them to the applicable registers.

During this phase, the internal STCU watchdog is started and is only cleared after completion of STCU configuration.

If self-test execution is the configuration information from the DCF record requests, the device moves to IDLE[DEST] and the STCU3 begins self-test. At the end of the test, the STCU3 asserts a functional interrupt causing the reset generation module to move to PHASE1[FUNC]. If self-test execution is not requested, the device moves directly to IDLE[FUNC]. The reset generation module determines whether self-test execution is enabled or not by examining an entry in the DCF record stored in the UTEST flash memory.

Note: When self-test execution is not requested, it is possible to maintain the device in PHASE3[DEST] by forcing RESETN pin low. This can be used for debugging purposes by allowing external test/development equipment to connect through JTAG to the internal debugger.

6.3.7 IDLE[DEST] phase: self-test execution

This phase is entered upon exit from PHASE3[DEST] if the execution of self-test is enabled.

During the IDLE[DEST] phase:

- Dedicated watchdog is configured to monitor self-test execution completion within the expected time. The length of this phase is variable depending on the self-test requirements.
- The self-test control unit (STCU3) executes all tests specified by the DCF information retrieved in PHASE3[DEST].
- The self-test engine updates the self-test completion flag and triggers an associated functional reset on completion of self-test.

The length of this phase is variable depending on the self-test requirements.

Note: When the SSCM is decoding the Device configuration field, it sets and clears various flags in STCU registers and memory that can later be examined by the SSCM to determine which self-test directives must be run.

Exiting IDLE[DEST] and moving to PHASE1[FUNC]:

- The IDLE[DEST] phase is automatically exited when the self-test code generates a functional reset on completion.
- The functional reset causes the reset generation module to move to the PHASE1[FUNC] phase.

Note: In the case of an unrecoverable fault detected during the execution of the offline BIST, the STCU requests that a destructive reset and device re-enters PHASE0[DEST].

In the case of a permanent fault, when the reset escalation is reached, the hardware keeps the device in PHASE0[DEST] until a power on reset occurs.

Whereas in the case of a transient fault, the software can read the cause of the reset by reading the RCC register flag DESR.D_STCU_CF.

6.3.8 PHASE1[FUNC] phase: temporization setup

This phase is entered either:

- On exit from IDLE[DEST].

- Immediately from PHASE2[FUNC], PHASE3[FUNC], or on IDLE[FUNC] when a nonmasked external or functional reset event occurs, providing the source of the reset event has not been configured to trigger a ‘short’ reset sequence.

During PHASE1[FUNC]:

- Temporization to allow correct NVM reset is applied.

To exit PHASE1[FUNC] and enter PHASE2 [FUNC]:

- All enabled, non-shortened functional resets must be processed.
- All processes started in PHASE1[FUNC] must be completed.

During this phase, test modules are available.

6.3.9 PHASE2[FUNC] phase: flash initial configuration

This phase is entered on exit from PHASE1[FUNC].

During the PHASE2[FUNC] phase:

- The reset signal is released to the flash memory.
- Flash initialization and configuration begins using a state machine that is internal to the flash memory module.
- A timeout watchdog inside RCC monitors flash initialization to ensure a reset in case of flash memory configuration failure.

The reset state machine exits PHASE2[FUNC] and enters PHASE3[FUNC] on verification that all processes started in PHASE2[FUNC] are completed.

6.3.10 PHASE3[FUNC] phase: device configuration monitoring

PHASE3[FUNC] is entered:

- On exit from PHASE2[FUNC].
- Immediately from IDLE[FUNC] when an enabled, short functional reset event occurs.

During PHASE3[FUNC], configuration information contained within the UTEST flash memory sector is checked against information extracted during PHASE3[DEST] and written to the Self-test control unit (STCU3). In case of mismatches, a dedicated boot fault is triggered within the FCCU that, if enabled, can cause a system reset. The purpose of PHASE3[FUNC] is to verify that the DCF information transferred to the STCU, RCC and other modules, in PHASE3[DEST], was performed correctly and has not been corrupted.

The reset generation module exits PHASE3[FUNC] and enters the IDLE[FUNC] on verification of the following:

- All processes started in PHASE3[FUNC] are completed.
- RESETN is not forced low externally.
- A minimum number of cycles has elapsed since the last enabled, shortened functional reset event.

After all PHASE3[FUNC] internal actions have been completed, it is possible to maintain the device within PHASE3[FUNC] by forcing RESETN pin low. This can be used for debugging purposes to connect through the JTAG port to the internal debugger.

6.3.11 IDLE[FUNC] phase

This is the reset generation module's final phase of the reset sequence. It is entered either on exit from PHASE3[DEST] (if self-test was not enabled) or at the completion of PHASE3[FUNC]. When the IDLE[FUNC] phase is reached, the RCC releases control of the system to the system status and control module. The SSCM, which started in PHASE3[FUNC], (or in PHASE3[DEST] in case no self-test was enabled) runs and waits for a signal from the RCC indicating that the reset state machine is now in the IDLE state. The RCC then waits for new reset events that can trigger a reset sequence.

6.3.12 System start-up

At the start-up, only one processor (from now on called boot CPU) and, if enabled, the Security module CPU wake up.

6.3.12.1 Boot CPU reset vector

After the reset phase is completed, the boot CPU starts executing at the value forced onto its reset vector. The reset vector is driven by the RCC peripheral, according to the value stored into the RCC register C1_VTOR_INIT_REG.

The RCC register C1_VTOR_INIT_REG for the boot CPU is initialized basing on life cycle and the system startup configuration data, stored in Flash and read by the SSCM from the Flash during the reset. Also the enabling of the Security module CPU is written by the SSCM, basing on the content of the related DCF record.

6.3.12.2 Boot CPU start-up

The boot CPU startup configuration comprises of the following settings:

- BAF_BYP configuration DCF bit inside Core1 Alternate Configuration DCF. It is used to decide whether the BAF code (stored by the manufacturer in the Flash BAF block) is skipped or executed. The customer Alternate BAF start address is also set within the same DCF.
- Core1 User VTOR Address DCF (programmed by the customer). It is used to set the customer application entry point after reset or after the BAF execution (if enabled).

The target for this device is to let the customer choose either the BAF code or directly the customer application software to be started after the reset phase using the same sales part number.

The Core1 Alternate Configuration DCF, that is used to set the BAF configuration, must be permanently written into the UTest block.

If the BAF_BYP configuration DCF bit is not programmed, the default value for the reset start address depends on the Life cycle.

If the BAF_BYP configuration DCF bit is programmed, its value determines whether the CPU starts directly from the BAF code (programmed by ST in the Flash BAF block) or at the Alternate BAF Address entry point defined by the customer.

The application entry point is retrieved in HW by the SSCM and written inside the RCC register C1_VTOR_ADDR_REG. This register is read by the BAF, to jump to Application start address, at the end of BAF execution itself. The BAF makes some checks about the validity of the Application address, loaded by SSCM inside RCC.

The RCC register C1_VTOR_ADDR_REG can be updated by the user for a maximum of 4 times during the device lifetime, writing the corresponding DFC record inside the flash.

The field UPDATES_COUNT inside the RCC register C1_VTOR_ADDR_REG reports how many times the register has been written by the SSCM during the reset sequence, indicating how many updates of the Application address has been stored inside NVM memory.

6.3.12.2.1 Start-up flow chart

The setting of the boot CPU start address is performed through a Hard Ware flow, based on the device life cycle and the value of some Device Configuration Format (DCF) records.

The flow followed by the Reset State machine to determine the boot CPU start address is depicted in *Figure 15: Boot CPU start-up sequence*.

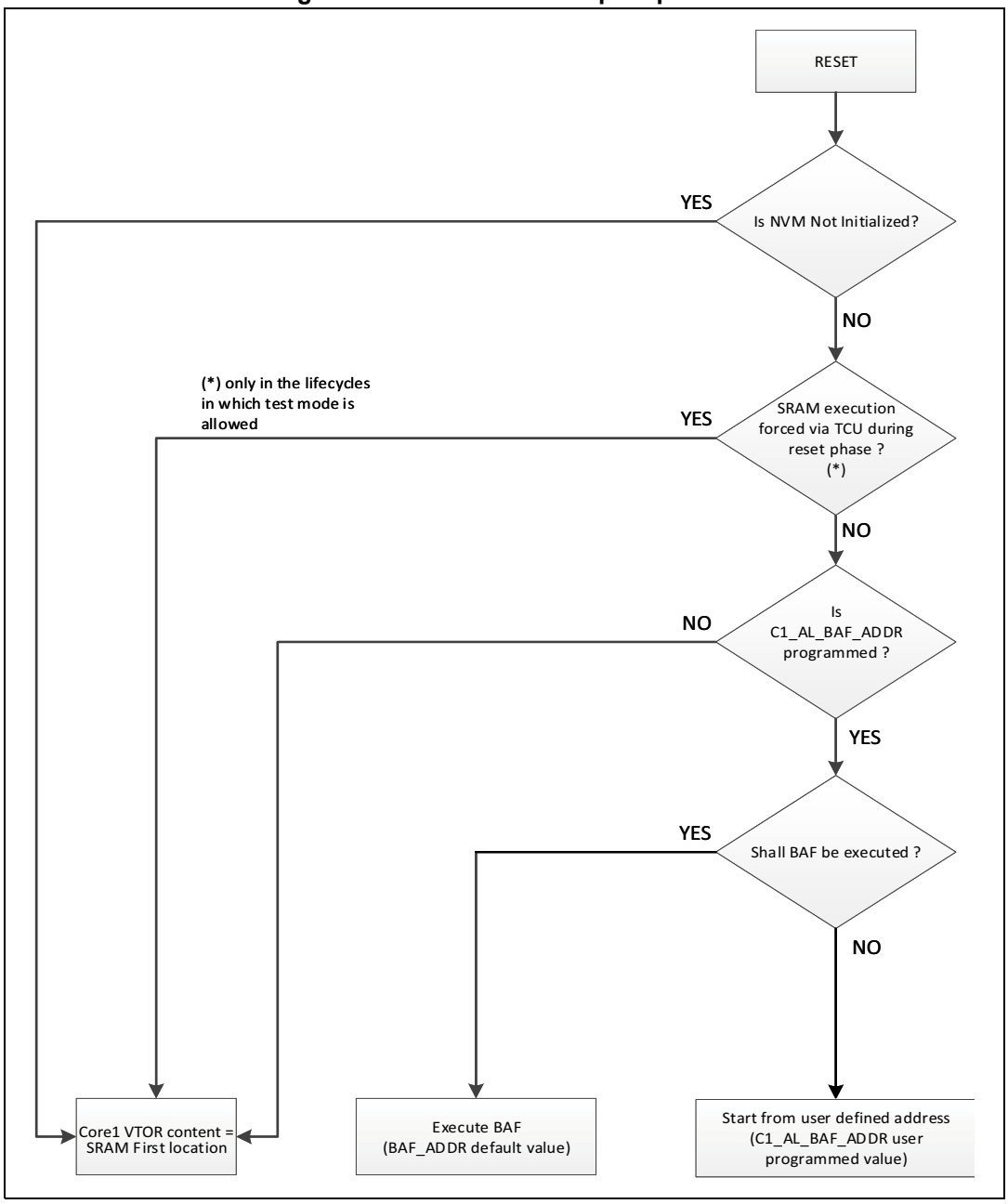
In case the NVM is not initialized or if the test mode is entered, the boot CPU reset vector is set at the start address of the System RAM (0x2400_0000h).

In case the BAF (Boot Assist Flash) is not bypassed, the boot CPU starts from the BAF default address (0x1FF0_0000), that contains the BAF code provided by STMicroelectronics device manufacturer.

In this case, at the end of BAF execution, a jump is done at the User Application Code start address, stored during boor phase inside the RCC register C1_VTOR_ADDR_REG.

In case the BAF is bypassed, the Boot CPU start address is set accordingly to the field C1_AL_VTOR_BAF_INIT of Core1 Alternate Configuration DCF record. The code at this address must be provided by the user and must take care of security and safety configurations of the device, in a similar way as the manufacturer BAF code.

Figure 15. Boot CPU start-up sequence



6.3.12.3 Security module CPU start-up

The Security module CPU is also woken up after a reset, if the hardware security module is enabled by DCF record. The security module CPU start address is at fixed location as it is not possible to provide a programmable Init Vector to the Security module CPU.

If HSM is not enabled to wake up in parallel to Boot CPU, it can be woken up also by Application SW, writing inside the RCC register HSM_SW_CTRLR. HSM SW enabling is conditioned by a Security DCF (See the security reference manual for additional information).

6.3.13 Waking-up the other CPU

The process of waking-up the other CPU (Core_2) is handled by writing some registers within the reset and clock control module (RCC) registers.

The start address of Core_2 CPU is loaded from NVM during the reset phase by SSCM.

The value of Core2 Vector Table Offset DCF is copied by SSCM inside the RCC register C2_VTOR_INIT_REG.

If the BAF code is executed, it reads from the RCC register SYS_CFG_REG if Core_2 has to be started. The Core_2 can be woken up by writing inside the RCC register C2_BOOT_CTRL_REG.

If the BAF code is not executed, it is the responsibility of the user code to properly write the RCC module registers and to wake up the other CPU from the intended addresses.

6.3.14 BAF serial bootloader

The BAF code includes a way to upload via CAN or UART some code into the system RAM and execute it. The possibility to execute the bootloader is conditioned by the life cycle status, which regulates what can or cannot be done according to the stage of the microcontroller. If the bootloader is allowed by the life cycle, then it is executed whenever the boot CPU executing code in the BAF does not find a valid content in the RCC register C1_VTOR_ADDR_REG. This register content is defined by the Core1 User VTOR Address DCF record. This mode uses a defined protocol to receive a program over a serial port via UART or CAN. The boot CPU then jumps to the first instruction of the uploaded program and begins execution.

The BAF bootloader code includes the following steps:

1. Configure UART or M_CAN module and external pins
2. Wait for START WORD
3. Receive START WORD
4. Receive START ADDRESS
5. Receive DOWNLOAD SIZE
6. Receive CODE
7. Branch to START ADDRESS

Note: There is a software watchdog timer that is started by the BAF code. If a program is not received before the watchdog timer timeout, a reset occurs.

6.4 RESETN pin functionality

The device has one reset pad:

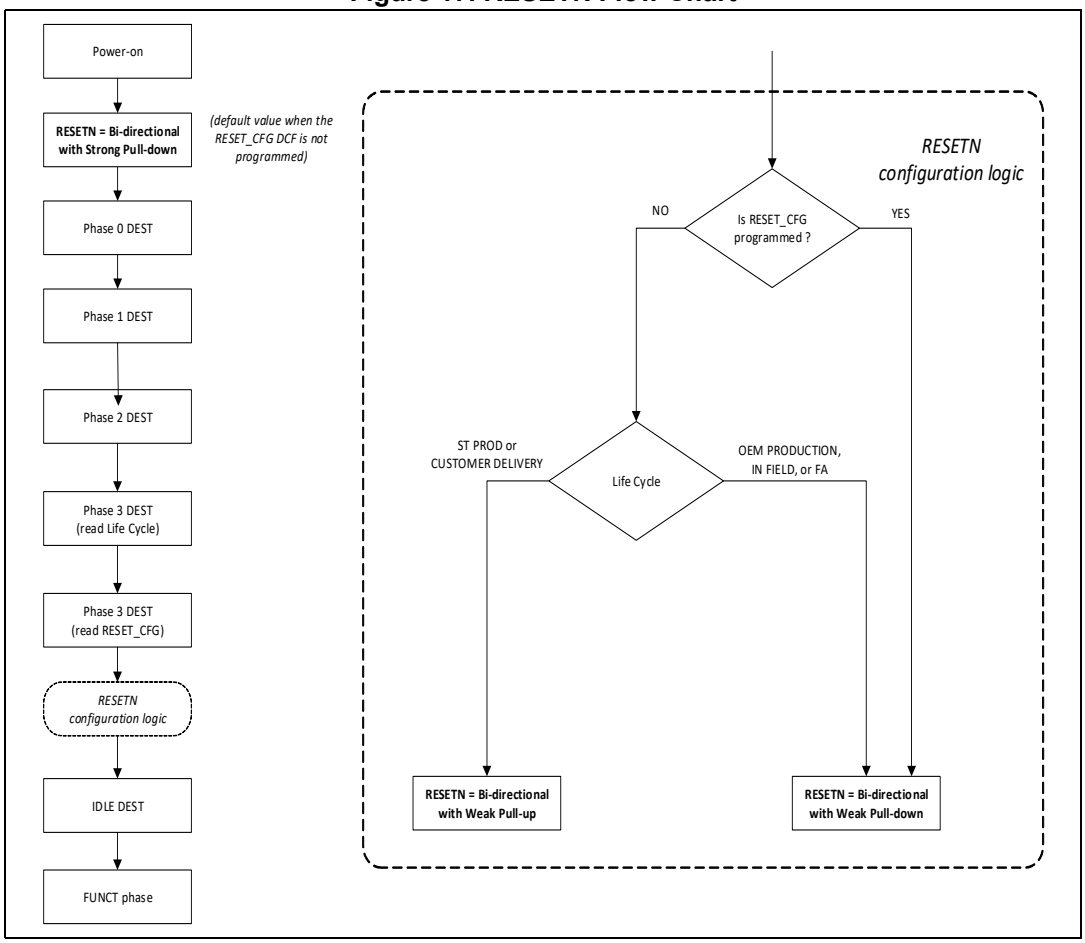
- RESETN: it can be configured as reset input/output.

The settings of RESETN pad are determined at device power-on and depend on life cycle and on RESET_CFG DCF, as shown in *Figure 16: RESETN Settings* and *Figure 17: RESETN Flow Chart*.

Figure 16. RESETN Settings

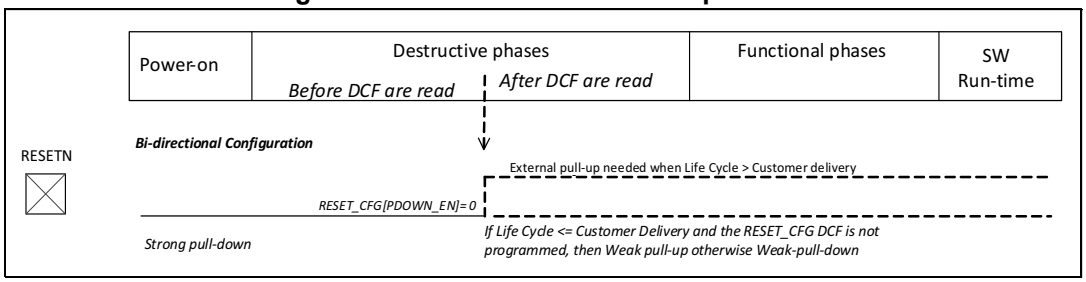
Input conditions		Behavior (set at power-on)		
Life Cycle	**DCF RESET_CFG [RESN_PDOWN_EN]** *Description : RESETN PullDown Enable*	**From Power on till DCFs are read**	**Any non-Power On reset phase (either Destructive or Functional) after internal RESETOUT release**	**SW Run time**
ST Production , Customer Delivery	*NOT programmed*	Strong pull -down	Bidirectional with Weak pull -up Pad level as forced from outside	
OEM Production , In-Field , Failure analysis	*NOT programmed*	Strong pull -down	Bidirectional with Weak pull -down Pad level as forced from outside	
Don't care	*Programmed to 0*	Strong pull -down	Bidirectional with Weak pull -down Pad level as forced from outside	

Figure 17. RESETN Flow Chart



The waveforms associated to RESETN configuration at Power-on are shown in *Figure 18: RESETN behavior set-up at POR*.

Figure 18. RESETN behavior set-up at POR



As RESETN pad is bidirectional, it has the capability to report in output the POR reset plus the enabled destructive and long functional resets.

The RESETN input buffer is always enabled.

When the RESETN pad reports in output the reset of the device generated by an internal source, the pad output buffer is enabled too, while the input from pad is masked towards the internal of the device.

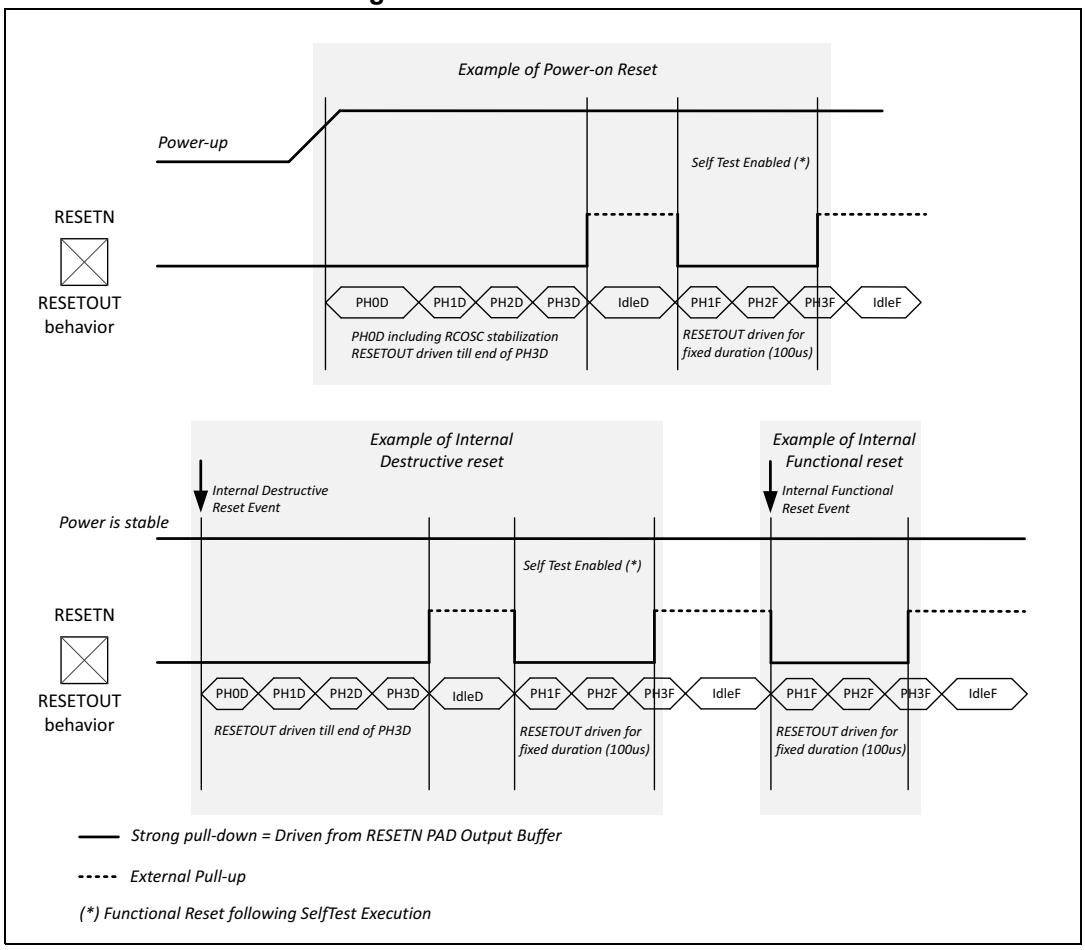
The functionality of reporting an internal reset in output is called RESETOUT and is identified also as “strong pull-down active” (because the output buffer of the pad is enabled).

The behavior of the RESETN pad, in case of RESETOUT, is depicted in *Figure 19: RESETOUT behavior* for the various type of internal resets: POR reset (always reported), destructive (always reported) and long functional resets (reported, if enabled).

When internal pull-down is enabled, the external pull-up has to be chosen to win the internal weak pull-down, and to allow to see the RESETOUT outside.

A transition from high to low of the RESETN pad has to trigger a destructive reset. Once the falling edge is detected, the actual value of RESETN pad has no effect till the end of RESET PHASE3, where a low value from outside or from internal weak pull-down can stretch the exit from PHASE3 and the overall reset duration.

Figure 19. RESETOUT behavior



The RESETN pad settings are regulated by RESET_CFG DCF.

RESET_CFG DCF client is reset by a power-on reset only.

This DCF has to be programmed to make effective the activation of the pull-down, otherwise the pull-up/pull-down configuration of the RESETN pad is regulated by the life cycle.

6.5 Reset source mapping

Table 88: Destructive Reset Sources shows the various destructive reset events and *Table 89: Functional Reset Sources* shows the various functional reset events.

The following symbols are used within the tables:

D: Feature is disabled and cannot be enabled

E: Feature is enabled and cannot be disabled

P[D]: Feature is programmable, default is set to: disabled

P[E]: Feature is programmable, default is set to: enabled

6.5.1 Destructive reset sources

This section describes the destructive reset sources.

Table 88. Destructive Reset Sources

Reset source		Configuration		
Name	**Description**	**Input**	**RESETN OUT(1)**	**Escalation**
D_POR	Device Power On-Reset (POR)	POR	E	—
D_RESETN	RESETN pin	D[1]	E	—
Reserved	Reserved	D[2]	—	—
D_RCC_SW	Software destructive reset from RCC	D[3]	E	—
D_FCCU	FOSU reset	D[4]	E	—
D_STCU_CF	STCU critical failure indicated by STCU_CF trigger(2)	D[5]	E	—
Reserved	Reserved	D[6:7]	—	—
D_EDR	Escalation destructive reset	D[8]	E	—
D_SWT_Boot(3)	Boot watchdog timeout during NVM initialization	D[9]	E	—
D_JTAG	JTAG destructive reset request	D[10]	P[E]	—
Reserved	Reserved	D[11:13]	—	—
D_SSCM	SSCM destructive reset	D[14]	E	—
D_HSM(4)	HSM destructive reset request	D[15]	E	—
D_TSENS	Temperature sensor	D[16]	E	—
Reserved	Reserved	D[17:23]	—	—
D_VD	Voltage out of range from LVDs and/or HVDs	D[24]	E	—
Reserved	Reserved	D[25:31]	—	—

1. This column shows how the RESETN reports in output the internal destructive reset, if reset source is enabled.
2. Same UF signal sent to the FCCU.
3. This trigger is generated by the watchdog only in case of timeout happening during the reset phase. When the reset phase is completed, this trigger is internally disabled by the RCC.
4. The possibility to trigger a destructive reset controlled by the HSM can be enabled or disabled via DCF.

6.5.2 Functional reset sources

This section describes the functional reset sources.

Table 89. Functional Reset Sources

Reset source		Configuration				
		Input	Functional Reset type (1)		RESETN OUT(2)	Escalation
Name	Description		Long Reset	Short Reset		
Reserved	Reserved	F[1:0]	—	—	—	—
F_ST_DONE	Self-test functional reset	F[2]	E	D	E	—
F_RCC_SW	Software functional reset from RCC	F[3]	P[E]	P[D]	P[D]	—
Reserved	Reserved	F[4]	—	—	—	—
F_FCCU_LONG	Long functional reset	F[5]	E	D	P[D]	—
F_FCCU_SHORT	Short functional reset	F[6]	D	E	D	—
Reserved	Reserved	F[7:9]	—	—	—	—
F_JTAG	JTAG reset	F[10]	P[E]	P[D]	P[D]	—
Reserved	Reserved	F[11:14]	—	—	—	—
F_HSM(3)	HSM Functional Reset	F[15]	E	P[D]	P[D]	—
F_TSENS	Temperature sensor	F[16]	P[E]	P[D]	P[D]	—
Reserved	Reserved	F[17:23]	—	—	—	—
F_VD	Voltage detector for Peripheral Domain	F[24]	P[E]	P[D]	P[D]	—
Reserved	Reserved	F[25:28]	—	—	—	—
Reserved	Reserved	D[30:31]	—	—	—	—

1. Set by specific RCC register (FORER register)
2. This column shows how the RESETN reports in output the internal functional reset: only enabled long functional resets are reported in output on RESETN pad. If the reset source is programmed to generate a short functional reset, the internal reset is not brought in output.
3. The possibility to trigger a functional reset controlled by the HSM can be enabled or disabled via DCF

6.5.3 Functional reset escalation

‘Functional' reset escalation can be used to generate a 'destructive' reset if a number of 'functional' resets, triggered by any of the functional reset input signals, have occurred between software writes to a specific RCC register.

This function is enabled by writing a non-zero value in a specific RCC register (FRETR register). Default value at reset of this register is 15.

Once 'functional' reset escalation has been enabled, the RCC increases a counter on each 'functional' or external reset, which causes a reset sequence to be initiated (that is an entrance into PHASE 1 or PHASE 3 from the IDLE phase). This counter is cleared on a write of any value to the specific RCC register and on any power-on or 'destructive' reset. If the counter reaches the value set into a specific RCC register, the RCC starts a destructive reset sequence at PHASE 0.

6.5.4 Destructive reset escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until a new power-on triggers a reset sequence if a number of 'destructive' resets, triggered by any of the destructive reset input signals, has occurred between software writes to a specific RCC register.

This function is enabled by writing a non-zero value to a specific RCC register (DRETR register). The default value at reset for this register is 15.

Once 'destructive' reset escalation has been enabled, the RCC increases a counter on each 'destructive' reset, which causes a reset sequence to be initiated (that is entrance into PHASE 0 from the IDLE phase) or an ongoing reset sequence to restart (that is entrance into PHASE 0 from PHASE 1, PHASE 2, or PHASE 3).

This counter is cleared on a write of any value in the specific RCC register and on any power-on reset. If the counter reaches the value set into the specific RCC register, the RCC enters reset PHASE 0 and stays there until the next power-on reset occurs.

6.6 RAM preservation upon reset

The content of some RAM modules of the device is preserved in case of:

- All long functional resets
- All short functional resets

RAM preservation is implemented for Core1 and Core2 ITCMs and DTCMs and for the System RAM.

7 Device configuration format (DCF) records

7.1 Introduction

Device configuration format (DCF) records are used to configure certain registers in the device during system boot while the reset signal is asserted. An individual DCF record is 64 bits long. It consists of a pointer to the location of a register internal to the device and the respective data.

There are two broad categories of DCF records: TEST DCF records and UTEST DCF records.

- TEST DCF records are programmed by the MCU manufacturer. They are used mainly to:
  - Program registers involved in trimming trip points for voltage comparators
  - Adjusting analog to digital voltage supplies
  - Trim oscillator frequencies
  - Enable RAM repair

Note: The TEST DCF records are programmed into TEST Flash during production and cannot be modified. TEST Flash is not visible to the user.

- UTEST DCF records can be either:
  - Factory programmed during production testing.
  - User programmed when application code is written into the Flash memory. User-supplied UTEST DCF records start at the next location in the UTEST memory map following the factory UTEST DCF records. The user-defined DCF records set up the initial memory map, define which tests are executed by the Self-Test Control Unit (STCU) during the boot sequence, enable the HSM, assign Flash memory blocks to specific tamper detect regions, assign memory blocks as OTP and assign Flash blocks to be associated with specific password groups.

Note: DCF records are HW processed by SSCM during reset.

Soft DCF records are SW processed by BAF code execution. Please see Section 18.3.5.1: 'Soft' DCF clients in *Chapter 18: Boot assist flash (BAF).*

System boot is a complex process requiring a considerable amount of initialization to take place before releasing reset. Before the device can be used in an application, the user application code, reset vectors for all of the CPUs and the DCF records must be properly programmed into their respective Flash memories.

7.2 DCF clients

DCF clients are 32-bit hardware registers inside a module that receive and store the data from a DCF record. This stored data is used to initialize registers and to configure features. DCF clients have a default value before any DCF records are written and may have special writing constraints like 'Write Once' or only allowing bits to be written from '1' to '0' or vice versa.

A list of DCF clients is shown in *Table 92*. (TEST DCF record only clients are not shown as these are not accessible by the user).

The DCF records select the target DCF client via a 30-bit field in the DCF record consisting of:

- A 15-bit chip-select field: each module that includes DCF clients is assigned a chip select during chip definition.
- A 15-bit address field: the address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

7.3 DCF records

DCF records are contiguous double-word (64 bits) entries programmed in a reserved area of OTP UTEST Flash memory beginning at 0x1FF8_0300. The ECC granularity is 64 bits (complete with a dummy DCF if needed). Address alignment is on 64 bits.

Caution: Once a stop record is detected, the SSCM stops scanning for further records.

The structure of a DCF record is shown in the figure below.

Bit n°	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	CS14	CS13	CS12	CS11	CS10	CS9	CS8	CS7	CS6	CS5	CS4	CS3	CS2	CS1	CS0	ADDR															PAR	STOP

Bit n°	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	DATA_MSW																DATA_LSW															

Figure 20. DCF record structure

Table 90. DCF record field description

Field	Description
63:49 CSn	Chip select n 1 Chip select is asserted 0 Chip select is negated **Note:** Only assert one bit per DCF record to select the target module for the DC client. All other chip selects must be negated.
48:34 ADDR	Address (client selection) The address field determines which client, within the target IP, has to receive the data.
33 PAR	DCF record parity This field is used to set even parity calculation for the data field DCF record. The DCF client reports an error for any calculated parity that does not match the value of this bit.
32 STOP	Stop bit This bit determines if the SSCM keeps on scanning the current DCF area beyond the current DCF or stop the reading. 0 SSCM keeps on scanning. 1 SSCM stops scanning and move to the next DCF area.
31:0 DATA_MSW DATA_LSW	Payload This field contains the DCF data, which is used by the client to drive its output signals.

The following structure must be present:

- The first record must be a start record.
- DCF records containing configuration data must immediately follow the start record with no blank records between an unprogrammed record is interpreted as a stop record and no DCF records following that record are processed.
- The end of the configuration records are indicated by the presence of a bit in the DCF record (stop record).

Each record is 64 bits long. *Figure 21* shows the DCF start record.

Caution: The start record must be placed at the beginning of the DCF area in UTEST Flash memory to indicate that the following data records must be processed.

Figure 21. DCF start record

0x00 0:31	0x04 32:63
0x05AA55AF	0x00000000

During system boot, while the reset signal is asserted, the SSCM reads the device configuration records and writes the data to the appropriate internal module registers. Many operational aspects of the device are therefore already configured when the reset signal is released and normal device operation begins.

Some device configuration records are calculated during production testing and programmed into the TEST Flash memory area. Other DCF records are developed by the user and programmed into the UTEST Flash area.

The stop bit designates the last device configuration record in a set of records. If it is set to ‘1’ within a DCF, then this DCF is the last one being read during the reset phase. A DCF can have the stop bit set to ‘1’ in two modes:

a) The Flash memory location of a DCF is not programmed (that is all the bits are set to ‘1’).

b) A DCF is programmed and the stop bit is set to ‘1’.

When the device is shipped from the factors, some DCF records are already programmed and the Flash memory location after the last DCF is left unprogrammed to make it possible to add additional DCF records.

A dummy DCF can be one of the following:

a) The same DCF record written within the 128 bits (that is the DCF record is written two times).

b) A DCF record with an invalid Chips Select (refer to the *Table 92* for a list of used Chip Selects).

The padded DCF must have the stop bit set to ‘0’ in order to be able to add new DCF records.

The general format of the stop record is shown in *Figure 22*. Only the stop bit needs to be ‘1’ in order to form a stop record, all other bits are ignored. An unprogrammed location in UTEST Flash is interpreted as a stop record.

Figure 22. DCF stop record

63:33	32	31:0
Reserved	1	Reserved

If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 91*.

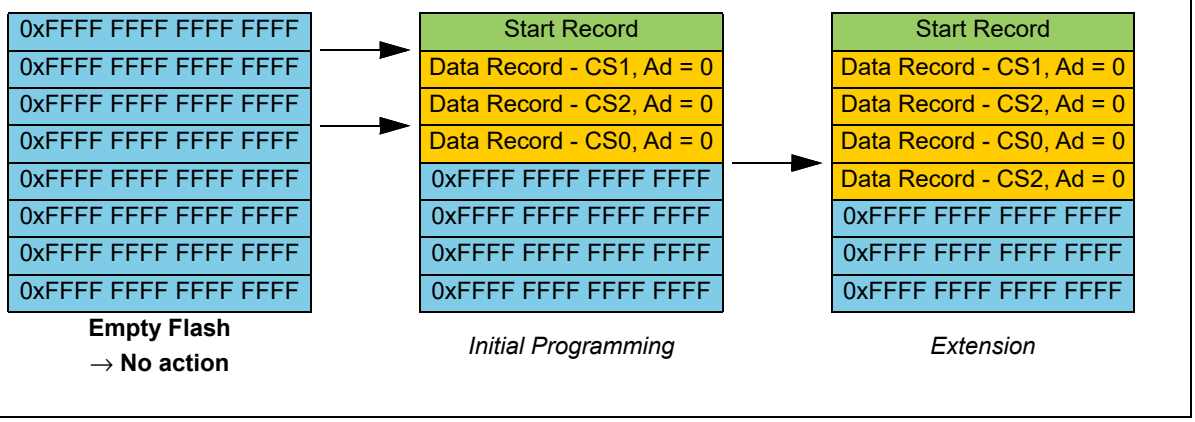
Table 91. Series of DCF records in UTEST Flash memory

ADDR offset	DATA			
0x00	0x05AA55AF			
0x04	0x00000000			STOP=0
0x08	WDATA[31:0]			
0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
0x10	WDATA[31:0]			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
8*n* + 0x0	Reserved			
8*n* + 0x4	Reserved			STOP=1
8(*n*+1) + 0x0	—			
8(*n*+1) + 0x4				

Caution: There must never be an unprogrammed record in the DCF data structure, as it is interpreted as a stop record and subsequent records are ignored.

Records programmed in several sessions are shown in *Figure 23*, appending new records to the end of the list each time.

Figure 23. Appending DCF records



It is possible for several DCF Records to write to the same DCF client. The later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

7.3.1 UTEST DCF records

UTEST DCF records are located in the UTEST Flash memory area (refer to UTEST Flash memory map in the Memory map chapter). Some UTEST DCF records are programmed at the factory and the user may add USER DCF records from the first unprogrammed location following the factory programmed UTEST DCF records.

When programming UTEST DCF records, the records must start at the first address in the UTEST Flash memory area and be continuous: one UTEST DCF record must immediately follow the previous UTEST DCF record. An unprogrammed location in the UTEST Flash memory area is interpreted by the SSCM as the end of the UTEST DCF records. When this happens, the SSCM passes control of the boot sequence to the RCC module.

7.4 DCF client table

A list of DCF clients is shown in *Table 92*. Various attributes of the DCF clients are listed in these tables:

- DCF Chip Select [14:0]: the CS field is a 15-bit field with each bit acting as a module select signal for the internal modules. Only one bit in this field must be set in a DCF record.
- DCF address [16:2]: the 15-bit ADDR field specifies an internal address for a DCF client. These addresses are only used by the SSCM when interpreting DCF records and writing data to specified modules.
- DCF client description: the name of the DCF clients.
- Reset value of DCF client: the reset value of DCF client before being written by the SSCM with the associated DCF record.
- IPS read: this column specifies whether or not a DCF client can be read by user software after the release of reset.
- DCF client special strategy: registers are designated as DCF clients if they need to be written with a DCF record. Other features that must be designated for DCF clients include different write strategies listed below. The following list defines the special strategies:
  - None: no special DCF strategy is used.
  - Write Once: a DCF client can only be written once. The DCF client ignores subsequent writes.
  - Triple Voted: DCF clients have three copies of the register. The SSCM writes to all three registers in a single write cycle. The outputs of the three registers are majority voted together to determine the correct data value. Triple voting allows for ‘bit-flip’ error tolerance without changing the DCF client output data.
  - Write 0 only: a bit in a DCF client can only be written from a logic 1 to a logic 0. An attempt to write a logic 1 is ignored.
  - Write 1 only: a bit in a DCF client can only be written from a logic 0 to a logic 1. An attempt to write a logic 0 is ignored.

- DCF order dependency in Flash memory

  DCF order dependency in Flash memory: this column details whether or not there are special conditions on the location of a particular DCF record in Flash memory.

7.4.1 DCF client list

Table 92. DCF client list

DCF CS [14:0] (Binary)	DCF address [16:10] (Binary)	DCF address [9:2] (Binary)	DCF client description	Reset value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in Flash
RCC							
000_0001_0000_0000	0000000	00000000	Core Enable and LS Configuration	0x00000000	No	Triple Voted+ chk_wr2+ wr_once	0
000_0001_0000_0000	0000000	00000010	Core2 Vector Table Offset	0x00000000	No	Triple Voted+ write up to 4 times	0
000_0001_0000_0000	0000000	00000011	Core1 User VTOR Address DCF	0x00000000	No	Triple Voted+ write up to 4 times	0
000_0001_0000_0000	0000000	00000100	Core1 Alternate Configuration DCF	0x08000000	No	Triple Voted+ wr_once	0
000_0001_0000_0000	0000000	00000101-00001111	Reserved				
000_0001_0000_0000	0000000	00010000	RESET_CFG DCF	0x00001000	No	Triple Voted	0
000_0001_0000_0000	0000000	00010001-00011111	Reserved				
NIC and System configuration							
000_0010_0000_0000	0000000	00100000	AXI_INIx_READ_ QOS_DCF0	0x20000000	No	Triple Voted+ wr_once	—
000_0010_0000_0000	0000000	00100001	AXI_INIx_READ_ QOS_DCF1	0x00001111	No	Triple Voted+ wr_once	—
000_0010_0000_0000	0000000	00100010-00100011	Reserved				

Table 92. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF address [16:10] (Binary)	DCF address [9:2] (Binary)	DCF client description	Reset value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in Flash
000_0010_0000_0000	0000000	00100100	AXI_INIx_WRITE_QOS_DCF0	0x20000000	No	Triple Voted+ wr_once	—
000_0010_0000_0000	0000000	00100101	AXI_INIx_WRITE_QOS_DCF1	0x00001111	No	Triple Voted+ wr_once	—
000_0010_0000_0000	0000000	00100110	AXI_WDG	0x00FFFFFF	No	Triple Voted+ wr_once	—
000_0010_0000_0000	0000000	00100111	REG_HWLOCK	0x00000000	No	Triple Voted+ wr_once	—
000_0010_0000_0000	0000000	00101000-11111111	Reserved				
STCU(1)							
000_0000_0000_0100	0000000	00000010	STCU_SKC	0x00000000	yes	None	First Record among STCU, Key1/Key2 dcf record sequence; Write Key2 every (1024-8) clock edges
000_0000_0000_0100	0000000	00000000	STCU_RUN	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00000100	STCU_CFG	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00000101	STCU_RUN_DELAY	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00000110	STCU_PLL_CFG	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00001111	STCU_WDG	0x0000FFFF	yes	None	None
000_0000_0000_0100	0000000	00010010	STCU_CRCE	0x00000000	yes	None	None

Table 92. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF address [16:10] (Binary)	DCF address [9:2] (Binary)	DCF client description	Reset value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in Flash
000_0000_0000_0100	0000000	00010101	STCU_ERR_FM	0x00000000	yes	None	None
000_0000_0000_0100	0000010	00010001	STCU_MB0_UFM0[31:0]	0x00000000	yes	None	None
000_0000_0000_0100	0000010	00010010	STCU_MB0_UFM1[31:0]	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000001	STCU_MB0_TM	0x00000000	yes	None	None
000_0000_0000_0100	0001011	11100100	STCU_CB_UFM	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00010000	STCU_MBIST0_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00010001	STCU_MBIST1_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00010010	STCU_MBIST2_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00010011	STCU_MBIST3_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00010100	STCU_MBIST4_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00010101	STCU_MBIST5_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00010110	STCU_MBIST6_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00010111	STCU_MBIST7_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00011000	STCU_MBIST8_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00011001	STCU_MBIST9_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00011010	STCU_MBIST10_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00011011	STCU_MBIST11_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00011100	STCU_MBIST12_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00011101	STCU_MBIST13_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00011110	STCU_MBIST14_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00011111	STCU_MBIST15_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00100000	STCU_MBIST16_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00100001	STCU_MBIST17_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00100010	STCU_MBIST18_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00100011	STCU_MBIST19_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00100100	STCU_MBIST20_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00100101	STCU_MBIST21_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00100110	STCU_MBIST22_PTR	0x00000000	yes	None	None

Table 92. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF address [16:10] (Binary)	DCF address [9:2] (Binary)	DCF client description	Reset value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in Flash
000_0000_0000_0100	0001100	00100111	STCU_MBIST23_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00101000	STCU_MBIST24_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00101001	STCU_MBIST25_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00101010	STCU_MBIST26_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00101011	STCU_MBIST27_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00101100	STCU_MBIST28_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00101101	STCU_MBIST29_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00101110	STCU_MBIST30_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00101111	STCU_MBIST31_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00110000	STCU_MBIST32_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00110001	STCU_MBIST33_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00110010	STCU_MBIST34_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00110011	STCU_MBIST35_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00110100	STCU_MBIST36_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00110101	STCU_MBIST37_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00110110	STCU_MBIST38_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00110111	STCU_MBIST39_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00111000	STCU_MBIST40_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00111001	STCU_MBIST41_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00111010	STCU_MBIST42_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00111011	STCU_MBIST43_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00111100	STCU_MBIST44_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00111101	STCU_MBIST45_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00111110	STCU_MBIST46_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	00111111	STCU_MBIST47_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	01000000	STCU_MBIST48_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	01000001	STCU_MBIST49_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	01000010	STCU_MBIST50_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	01000011	STCU_MBIST51_PTR	0x00000000	yes	None	None

Table 92. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF address [16:10] (Binary)	DCF address [9:2] (Binary)	DCF client description	Reset value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in Flash
000_0000_0000_0100	0001100	01000100	STCU_MBIST52_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	01000101	STCU_MBIST53_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	01000110	STCU_MBIST54_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	01000111	STCU_MBIST55_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	01001000	STCU_MBIST56_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0001100	01001001	STCU_MBIST57_PTR	0x00000000	yes	None	None
000_0000_0000_0100	0010000	00100000	STCU_CBIST0_PTR	0x00000001	yes	None	None
000_0000_0000_0100	0010000	00100001	STCU_CBIST1_PTR	0x00000002	yes	None	None
000_0000_0000_0100	0010000	00100010	STCU_CBIST2_PTR	0x00000003	yes	None	None
000_0000_0000_0100	0010000	00100011	STCU_CBIST3_PTR	0x00000004	yes	None	None
000_0000_0000_0100	0010000	00100100	STCU_CBIST4_PTR	0x00000005	yes	None	None
000_0000_0000_0100	0010000	00100101	STCU_CBIST5_PTR	0x00000006	yes	None	None
000_0000_0000_0100	0010000	00100110	STCU_CBIST6_PTR	0x00000007	yes	None	None
000_0000_0000_0100	0010000	00100111	STCU_CBIST7_PTR	0x00000008	yes	None	None
000_0000_0000_0100	0010000	00101000	STCU_CBIST8_PTR	0x00000009	yes	None	None
000_0000_0000_0100	0010000	00101001	STCU_CBIST9_PTR	0x00000010	yes	None	None
PASS							
000_0000_0000_1000	0000000	00101000-00101011	Reserved				
000_0000_0000_1000	0000000	00101100	Censorship	0x0	Yes	None	None
000_0000_0000_1000	0000000	00101101	Password Slot Access	0x0	Yes	wr_once	None
000_0000_0000_1000	0000000	00101110-00101111	Reserved				
000_0000_0000_1000	0000000	00110000	Production Disable	0x0	Yes	wr_once	None
000_0000_0000_1000	0000000	00110001-00110111	Reserved				
000_0000_0000_1000	0000000	00111000	Fuse Bypass Enable	0x1	Yes	wr_once	None
000_0000_0000_1000	0000000	00111001-00111111	Reserved				
000_0000_0000_1000	0000000	01000000	LOCK0_PG0	0xFFFFFFFF	No	None	None

Table 92. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF address [16:10] (Binary)	DCF address [9:2] (Binary)	DCF client description	Reset value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in Flash
000_0000_0000_1000	0000000	01000001	LOCK1_PG0	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01000010	LOCK2_PG0	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01000011	LOCK3_PG0	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01000100	LOCK0_PG1	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01000101	LOCK1_PG1	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01000110	LOCK2_PG1	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01000111	LOCK3_PG1	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01001000	LOCK0_PG2	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01001001	LOCK1_PG2	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01001010	LOCK2_PG2	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01001011	LOCK3_PG2	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01001100	LOCK0_PG3	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01001101	LOCK1_PG3	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01001110	LOCK2_PG3	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01001111	LOCK3_PG3	0xFFFFFFFF	No	None	None
000_0000_0000_1000	0000000	01001100-11111111	Reserved				
Tamper Detect							
000_0000_0001_0000	0000000	00000000	Diary Base Address	0xFFFFFFFF	No	wr_once+ wr0_only	None
000_0000_0001_0000	0000000	00000001-00000111	Reserved				
000_0000_0001_0000	0000000	00001000	OTP_EN0	0x00000000	No	Triple Voted+ wr1_only	None
000_0000_0001_0000	0000000	00001001	OTP_EN1	0x00000000	No	Triple Voted+ wr1_only	None
000_0000_0001_0000	0000000	00001010	OTP_EN2	0x00000000	No	Triple Voted+ wr1_only	None

Table 92. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF address [16:10] (Binary)	DCF address [9:2] (Binary)	DCF client description	Reset value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in Flash
000_0000_0001_0000	0000000	00001011	OTP_EN3	0x00000000	No	Triple Voted+ wr1_only	None
000_0000_0001_0000	0000000	00001100-00010011	Reserved				
000_0000_0001_0000	0000000	00010100	TDR0_LOCK0	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00010101	TDR0_LOCK1	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00010110	TDR0_LOCK2	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00010111	TDR0_LOCK3	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00011000	TDR1_LOCK0	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00011001	TDR1_LOCK1	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00011010	TDR1_LOCK2	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00011011	TDR1_LOCK3	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00011100	TDR2_LOCK0	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00011101	TDR2_LOCK1	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00011110	TDR2_LOCK2	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00011111	TDR2_LOCK3	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00100000	TDR3_LOCK0	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00100001	TDR3_LOCK1	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00100010	TDR3_LOCK2	0x00000000	No	wr_once+ wr1_only	None

Table 92. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF address [16:10] (Binary)	DCF address [9:2] (Binary)	DCF client description	Reset value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in Flash
000_0000_0001_0000	0000000	00100011	TDR3_LOCK3	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00100100	TDR4_LOCK0	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00100101	TDR4_LOCK1	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00100110	TDR4_LOCK2	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00100111	TDR4_LOCK3	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00101000	TDR5_LOCK0	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00101001	TDR5_LOCK1	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00101010	TDR5_LOCK2	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00101011	TDR5_LOCK3	0x00000000	No	wr_once+ wr1_only	None
000_0000_0001_0000	0000000	00101100-11111111	Reserved				
MISCELLANEOUS							
000_0000_1000_0000	0000000	00000000	UTEST Miscellaneous	0x00010C76	No	Triple Voted	None
000_0000_1000_0000	0000000	00000010-00000110	Reserved				
000_0000_1000_0000	0000000	00000111	JTAG Pin Config	0x0000000E	No	None	None
000_0000_1000_0000	0000000	00001000-01000111	Reserved				
000_0000_1000_0000	0000000	01001001	PMC REE_BUS	0x00000000	No	Triple Voted	None
000_0000_1000_0000	0000000	01001011-11111110	Reserved				
000_0000_1000_0000	0000001-1111111	00000000-11111111	Reserved				

Table 92. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF address [16:10] (Binary)	DCF address [9:2] (Binary)	DCF client description	Reset value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in Flash
SOFT DCF CLIENTS (BAF)							
100_0000_0000_0000	0000000	00000000-11111111	Reserved				
100_0000_0000_0000	0000001	00000000-11111111	Reserved				
100_0000_0000_0000	0000010	00000000	ADDR	—	—	—	None
100_0000_0000_0000	0000010	00000001	DATA32	—	—	—	None
100_0000_0000_0000	0000010	00000010	DATA16	—	—	—	None
100_0000_0000_0000	0000010	00000011	DATA8	—	—	—	None
100_0000_0000_0000	0000010	00000100-00001111	Reserved				
100_0000_0000_0000	0000010	00010000	CALLBACK	—	—	—	None
100_0000_0000_0000	0000010	00010001	USER_SERIAL_BOOT	—	—	—	None
100_0000_0000_0000	0000010	00010001-11111111	Reserved				
100_0000_0000_0000	0000011-1111111	00000000-11111111	Reserved				

1. The STCU DCF description is aligned with the register description in the STCU3 chapter.

7.4.2 Miscellaneous DCF records

Offset: 00000000 Access: DCF Client

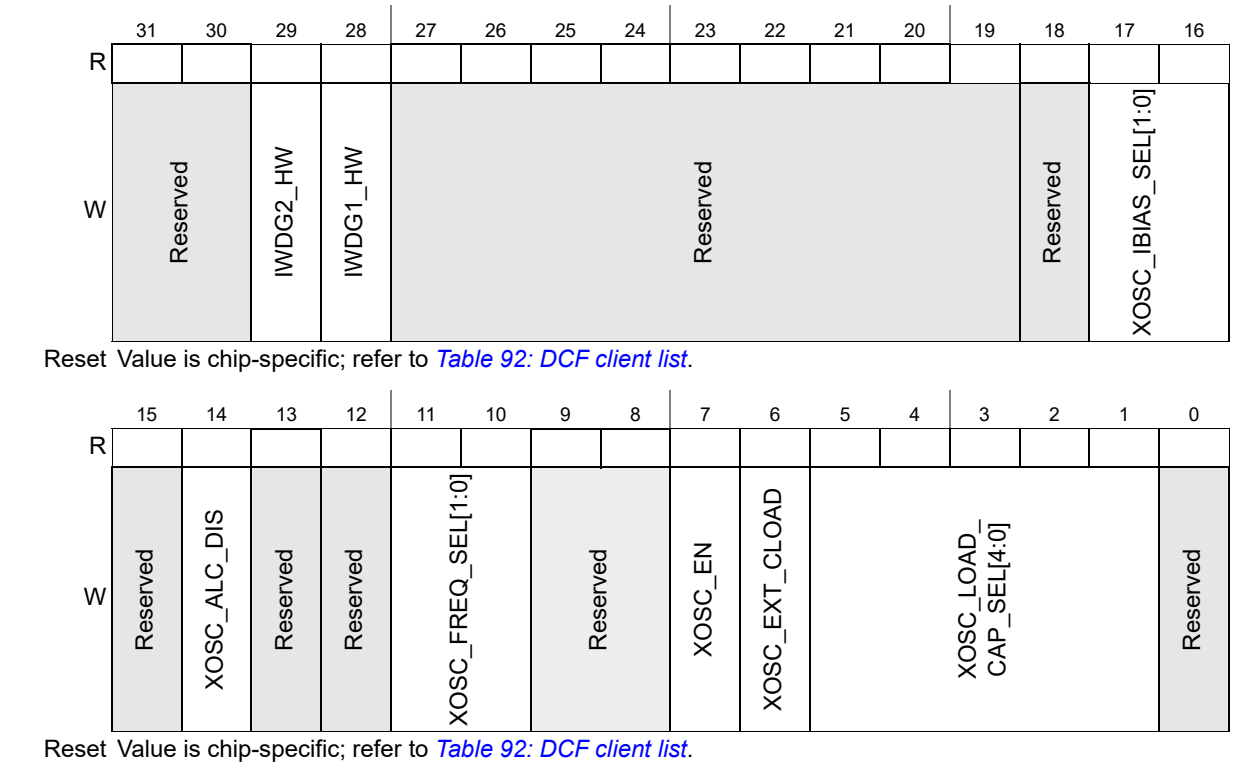


Figure 24. UTEST Miscellaneous DCF Client

Table 93. UTEST Miscellaneous DCF Client field descriptions

Field	Description
29 IWDG2_HW	1: IWDG2 is enabled by HW after reset 0: IWDG2 is disabled by HW after reset (SW to enable it)
28 IWDG1_HW	1: IWDG1 is enabled by HW after reset 0: IWDG1 is disabled by HW after reset (SW to enable it)
17:16 XOSC_IBIAS_SEL[1:0]	Allows to change oscillation margin (default “01”) - User to keep default value of “01”
14 XOSC_ALC_DIS	Active high to disable Automatic Level Controller
11:10 XOSC_FREQ_SEL[1:0]	This selects the different frequency settings of XOSC. 00 4 MHz – 10 MHz 01 10 MHz – 20 MHz 10 20 MHz – 30 MHz 11 30 MHz – 40 MHz
7 XOSC_EN	Enable XOSC 0 XOSC disabled 1 XOSC enabled

Table 93. UTEST Miscellaneous DCF Client field descriptions (continued)

Field	Description
6 XOSC_EXT_CLOAD	XOSC EXT CLOAD 0 Selects XOSC internal cap (recommended for 20 MHz – 40 MHz operation) 1 Selects XOSC external cap (recommended for 4 MHz – 20 MHz operation) **Note:** It is always recommended to verify the required size of the capacitances with the crystal supplier.
5:1 XOSC_LOAD_CAP_SEL[0:4]	XOSC LOAD CAP SEL [0:4] Five trim bits program the load capacitance. The formula to calculate the capacitance offered is $C_{tot} = C_{fix}+n.C_u$ where: Fixed Capacitance C_{fix} = 11.0pF (variation ±15%) Unit capacitance C_u = 0.49pF (variation ±15%) n = load_cap_sel[4] * 2^4 + load_cap_sel[3] * 2^3 + load_cap_sel[2] * 2^2 + load_cap_sel[1] * 2^1 + load_cap_sel[0] * 2^0

Offset: 00000111 Access: DCF Client

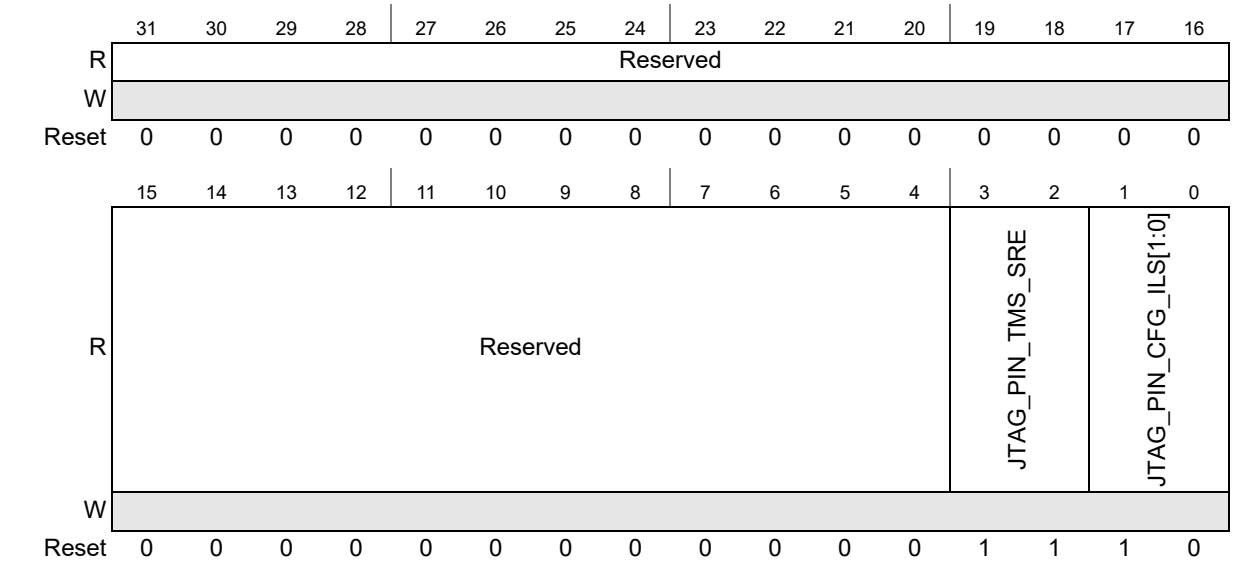


Figure 25. JTAG Pin Configuration

Table 94. JTAG Pin Configuration field descriptions

Field	Description
3:2 JTAG_PIN_TMS_SRE	JTAG pins slew rate control 00 slow 01 medium 10 fast 11 very fast
1:0 JTAG_PIN_CFG_ILS	JTAG pins mode 00 JTAG pins configured for CMOS mode 01 JTAG pins configured for AUTO mode 10 JTAG pins configured for TTL mode 11 JTAG pins configured for TTL mode Default: JTAG pins configured for TTL mode

Offset: 01001001 Access: DCF Client

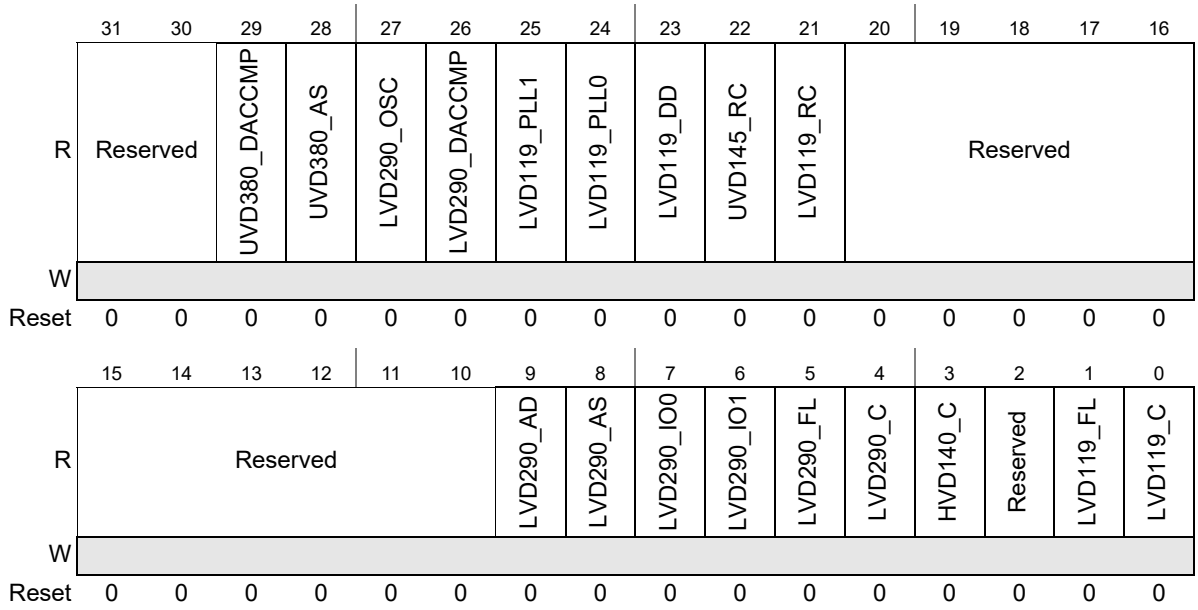


Figure 26. PMC_REE_BUS

Table 95. PMC_REE_BUS field descriptions

Field	Description
0 LVD119_C	This bit defines whether an LVD assertion on the core low voltage supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
1 LVD119_FL	This bit defines whether an LVD assertion on the flash low voltage supply generates a system reset. 0: Reset event disabled 1: Reset event enabled

Table 95. PMC_REE_BUS field descriptions (continued)

Field	Description
3 HVD140_C	This bit defines whether an HVD assertion on the core low voltage supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
4 LVD290_C	This bit defines whether an LVD assertion on the core high voltage supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
5 LVD290_FL	This bit defines whether an LVD assertion on the flash low voltage supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
6 LVD290_IO1	This bit defines whether an LVD assertion on the I/O supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
7 LVD290_IO0	This bit defines whether an LVD assertion on the I/O supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
8 LVD290_AS	This bit defines whether an LVD assertion on the SAR ADCs supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
9 LVD290_AD	This bit defines whether an LVD assertion on the SD ADCs supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
21 LVD119_RC	This bit defines whether an LVD assertion on the IRCOSC supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
22 UVD145_RC	This bit defines whether an HVD assertion on the IRCOSC supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
23 LVD119_DD	This bit defines whether an LVD assertion on the HRTimers' DLL supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
24 LVD119_PLL0	This bit defines whether an LVD assertion on the PLL0 supply generates a system reset. 0: Reset event disabled 1: Reset event enabled

Table 95. PMC_REE_BUS field descriptions (continued)

Field	Description
25 LVD119_PLL1	This bit defines whether an LVD assertion on the PLL1 supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
26 LVD290_DACCMP	This bit defines whether an LVD assertion on the DACs and Comparators supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
27 LVD290_OSC	This bit defines whether an LVD assertion on the XOSC supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
28 UVD380_AS	This bit defines whether an HVD assertion on the SAR-ADCs supply generates a system reset. 0: Reset event disabled 1: Reset event enabled
29 UVD380_DACCMP	This bit defines whether an HVD assertion on the DACs and Comparators supply generates a system reset. 0: Reset event disabled 1: Reset event enabled

Offset: 000000000000000 Access: DCF Client

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved														eFE	FTMPL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 27. Flash test mode protection

Table 96. Flash test mode protection field descriptions

Field	Description
30 eFE	0 Efuse system is disabled. 1 Efuse system is enabled. Default: Efuse system is enabled.
31 FTMPL	0 FTMP is not locked. 1 FTMP is locked. Default: FTMP is not locked.

7.4.3 RCC DCF records

Global System Configuration DCF

This DCF is used to configure:

- Cores to be started by hardware upon reset
- Lockstep configuration of the cores and DMAs
- Flash OTA mode

Access: DCF Client

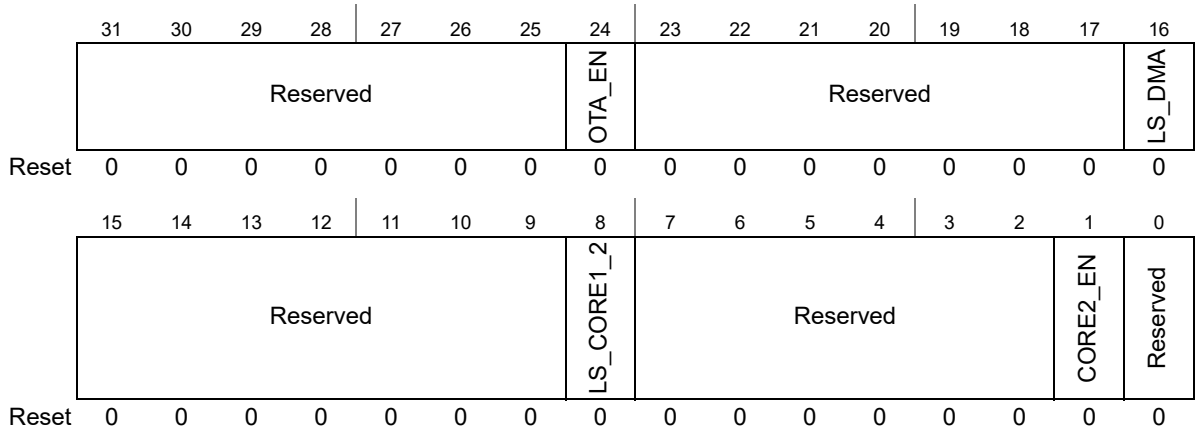


Figure 28. Global System Configuration

Table 97. Global System Configuration field descriptions

Field	Description
31:25	Reserved
24 OTA_EN	Enables OTA on flash 0: OTA disabled (default) 1: OTA enabled
23:17	Reserved
16 LS_DMA	Enables lockstep replicas for DMA1, DMA2 and DMAMUX 0: Lockstep for DMA1, DMA2 and DMAMUX disabled (default) 1: Lockstep for DMA1, DMA2 and DMAMUX enabled
15:9	Reserved
8 LS_CORE1_2	Enables lockstep configuration for Core1 and 2 0: Lockstep for Core1 and Core2 disabled - Split Configuration (default) 1: Lockstep for Core1 and Core2 enabled - Lock configuration **Note:** : Only C1_VTOR DCF has to be considered as starting address
7:2	Reserved
1 CORE2_EN	Core2 enabled 0: Core2 is not enabled 1: Core2 is enabled by HW and started by using C2_VTOR DCF data
0	Reserved

Core2 Vector Table Offset DCF

The C2_VTOR DCF configures the address of vector table of the core2 after reset release.

Access: DCF Client

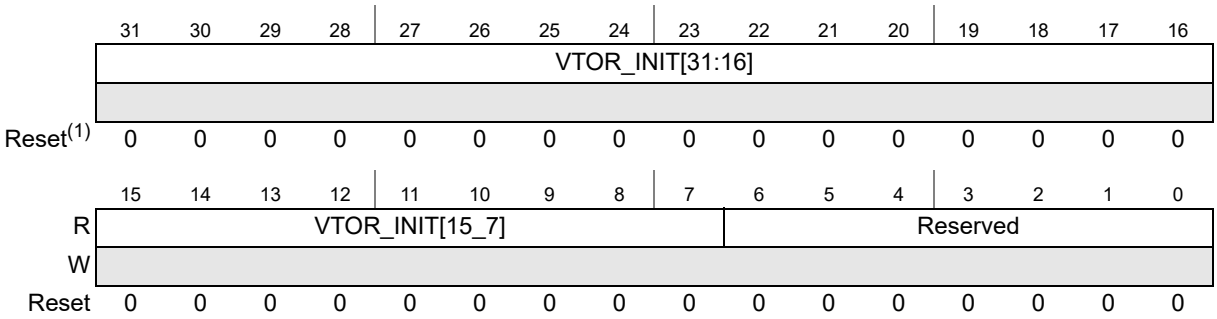


Figure 29. Core2 Vector Table Offset

1. Alias for ITCM of Cortex®-M7

Table 98. Core2 Vector Table Offset field descriptions

Field	Description
31:7 VTOR_INIT	Address of vector table of the M7 Core2 after reset release.
6:0	Reserved

The associated DCF client is triple voted and can be overwritten up to 4 times, allowing to update the VTOR_INIT address for a maximum of 4 times.

Core1 User VTOR Address DCF

This DCF defines the address of the vector table of the core1 after BAF is executed. The DCF content is mirrored as the reset value of the RCC register C1_VTOR_ADDR_REG.

Access: DCF Client

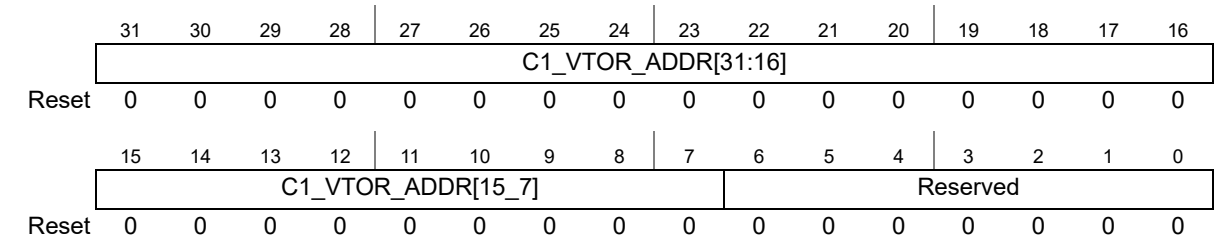


Figure 30. Core1 User VTOR Address

Table 99. Core1 User VTOR Address field description

Field	Description
31:7 C1_VTOR_ADDR	Address of core1 vector table for user application.
6:0	Reserved

The associated DCF client is triple voted and can be overwritten up to 4 times, allowing to update the Core1 Application address for a maximum of 4 times.

Core1 Alternate Configuration DCF

In case the User wants to bypass the BAF execution, this DCF has to be programmed. The Boot CPU starts execution from this user defined address.

Access: DCF Client

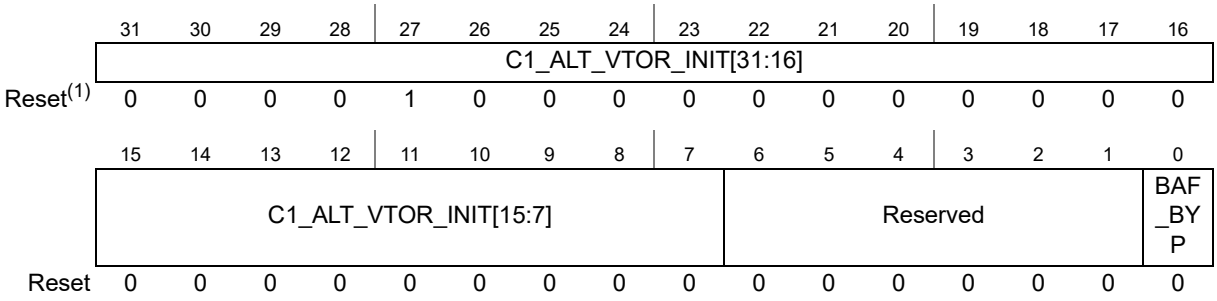


Figure 31. Core1 Alternate Configuration

1. Flash1 start address

Table 100. Core1 Alternate Configuration field description

Field	Description
31:7 C1_ALT_VTOR_INIT	Address of vector table of the Core1 if BAF_BYP bit is programmed.
6:1	Reserved
0 BAF_BYP	0: BAF is not bypassed 1: BAF is bypassed and the new address for boot core start is given by C1_ALT_VTOR_INIT field

This DCF is mirrored as reset value of RCC register C1_VTOR_INIT register when BAF_BYP bit is set to 1.

This DCF is write-once, triple voted and checked at second write during functional reset.

RESET_CFG_DCF

This DCF configures the activation of the pull-down of the RESETN pad, otherwise the pull-up/pull-down configuration is regulated by the life cycle.

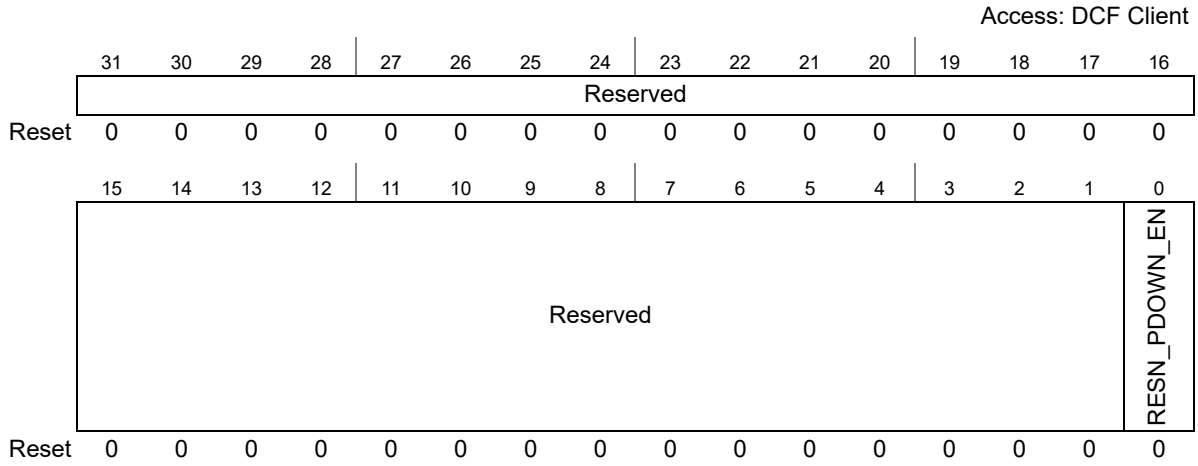


Figure 32. RESET_CFG DCF

Table 101. RESET_CFG DCF field description

Field	Description
31:1	Reserved
0 RESN_PDOWN_EN	0: This bit has to be programmed to 0 to make effective the enable of the Pull down on RESETN PAD basing on Life Cycle 1: No effect

REG_HWLOCK_DCF

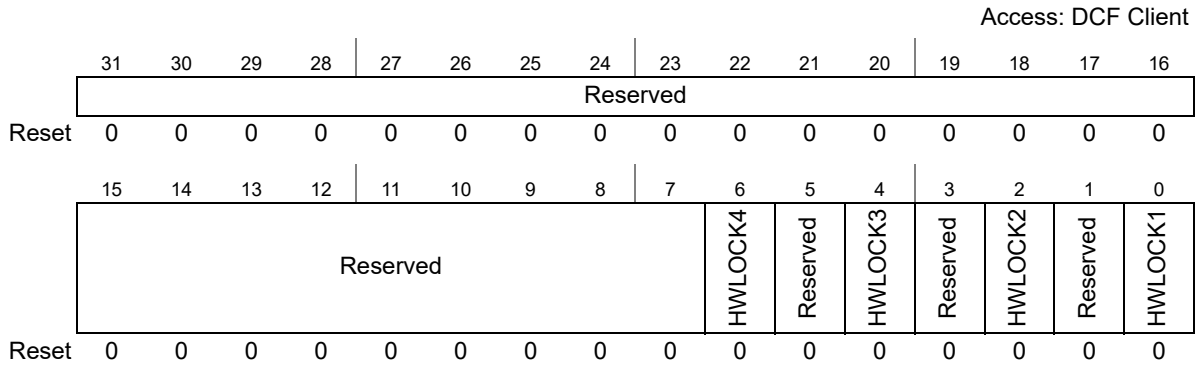


Figure 33. REG_HWLOCK_DCF

Table 102. REG_HWLOCK_DCF field description

Field	Description
31:7	Reserved
6 HWLOCK4	0: HW Lock register protection for APB Bridge2 not enabled 1: HW Lock register protection for APB Bridge2 enabled
5	Reserved

Table 102. REG_HWLOCK_DCF field description (continued)

Field	Description
4 HWLOCK3	0: HW Lock register protection for APB Bridge1 not enabled 1: HW Lock register protection for APB Bridge1 enabled
3	Reserved
2 HWLOCK2	0: HW Lock register protection for AHB Bridge2 not enabled 1: HW Lock register protection for AHB Bridge2 enabled
1	Reserved
0 HWLOCK1	0: HW Lock register protection for AHB Bridge1 not enabled 1: HW Lock register protection for AHB Bridge1 enabled

8 Power management

8.1 Overview

SR5E1x microcontrollers include a robust power management infrastructure that enables applications to monitor internal voltages for high- and low-voltage conditions. The monitoring capability is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller can leave reset. This chapter gives an overview of the built-in power management features.

The power management infrastructure comprises the following modules:

- Reset and Clock Controller (RCC)
- Power Management Controller (PMC)
- Power Management Controller Digital Interface (PMC_Dig)

This chapter describes these modules in brief. For complete details, see the respective module chapters.

The RCC generates reference clocks for all the chip blocks. It works in conjunction with internal clock gating logic to implement low power modes. RCC controls chip operational modes and mode transition sequences. It also contains configuration, control and status registers accessible for the application.

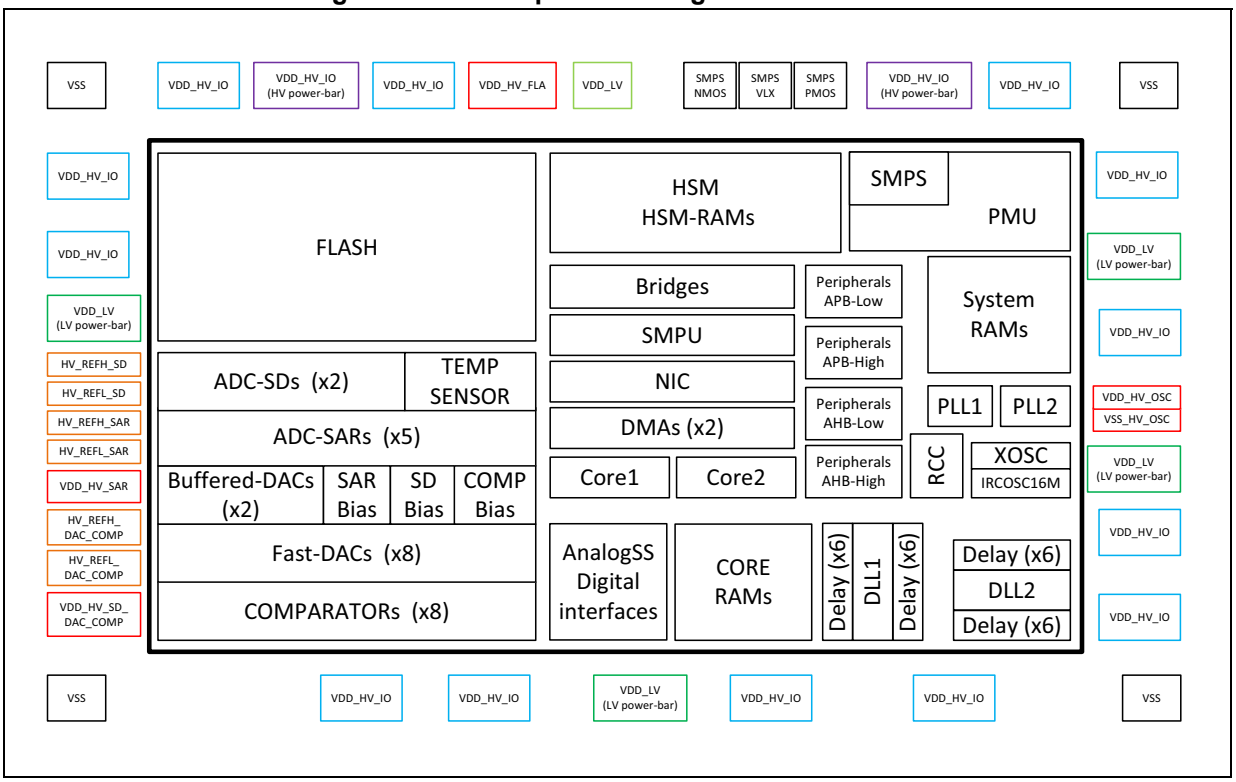
The PMC module contains all the internal voltage regulator and monitors of the device: refer to *Section 8.1.3: Power management controller overview*.

The PMC_Dig module is the digital interface of the PMC, and contains all the registers to control the PMC during reset and power mode changes.

8.1.1 Power management framework

The power management framework supports a variety of configuration options. See *Figure 34: Device power management framework* for more details. SR5E1x implements a unique core voltage power domain. There are no separate functional power domains.

Figure 34. Device power management framework



8.1.2 Power management supply description

The following table provides an overview of the PMC supply signals.

Table 103. Device power management controller external signals

Name	Type	Description
HV_REFH_SD	Reference	Voltage reference of ADC Σ/Δ module
HV_REFL_SD	Reference	Ground reference of ADC Σ/Δ modules
HV_REFH_SAR	Reference	Voltage reference of ADC SAR module
HV_REFL_SAR	Reference	Ground reference of ADC SAR modules
HV_REFH_DAC_COMP	Reference	Voltage reference of all Fast-DAC, for all Buffered-DAC and for all COMPARATOR modules
HV_REFL_DAC_COMP	Reference	Ground reference of all Fast-DAC, for all Buffered-DAC and for all COMPARATOR modules
$V_{DD_HV_SD_DAC_COMP}$	Supply	High Voltage power supply for all ADC-SD, DAC and COPARATOR modules
$V_{DD_HV_SAR}$	Supply	High voltage supply for the ADC-SAR modules
$V_{DD_HV_FLA}$	Supply	Supply pin for Flash
$V_{DD_HV_IO}$	Supply	High voltage power supply for I/O and for PMU/SMPS
V_{DD_LV}	Supply	Low voltage power supply for the core area generated by internal regulator by SMPS circuitary and external FET

Table 103. Device power management controller external signals (continued)

Name	Type	Description
V_{SS}	Ground	Ground supply for the device / I/O. This is covering both VSS_LV and VSS_HV (exposed pad device)
$V_{DD_HV_OSC}$	Supply	High Voltage power supply for Oscillator
$V_{SS_HV_OSC}$	Supply	Ground supply for Oscillator

Caution: There are constraints on some input voltages relative to other input voltages. Please refer to the device data sheet for detailed information.

8.1.3 Power management controller overview

The power management controller unit (PMC) contains the device internal voltage regulator and all the voltage monitors. The power management unit manages the power requirements of the device by generating and monitoring regulated voltage required for the correct operation of the device.

The power management unit performs many functions, like:

- Correct power up /power down of the device,
- Generate the regulated voltage (1.28 V) for core by switching regulator, which is part of the PMC IP,
- Monitor multiple supplies.

Refer to the data sheet for the electrical characteristics of the regulators and the monitors and the list of external components required. The list of functionalities of the monitors is described further in this chapter.

The following figure describes the top level block diagram of power management unit.

Figure 35. Power management controller block diagram overview

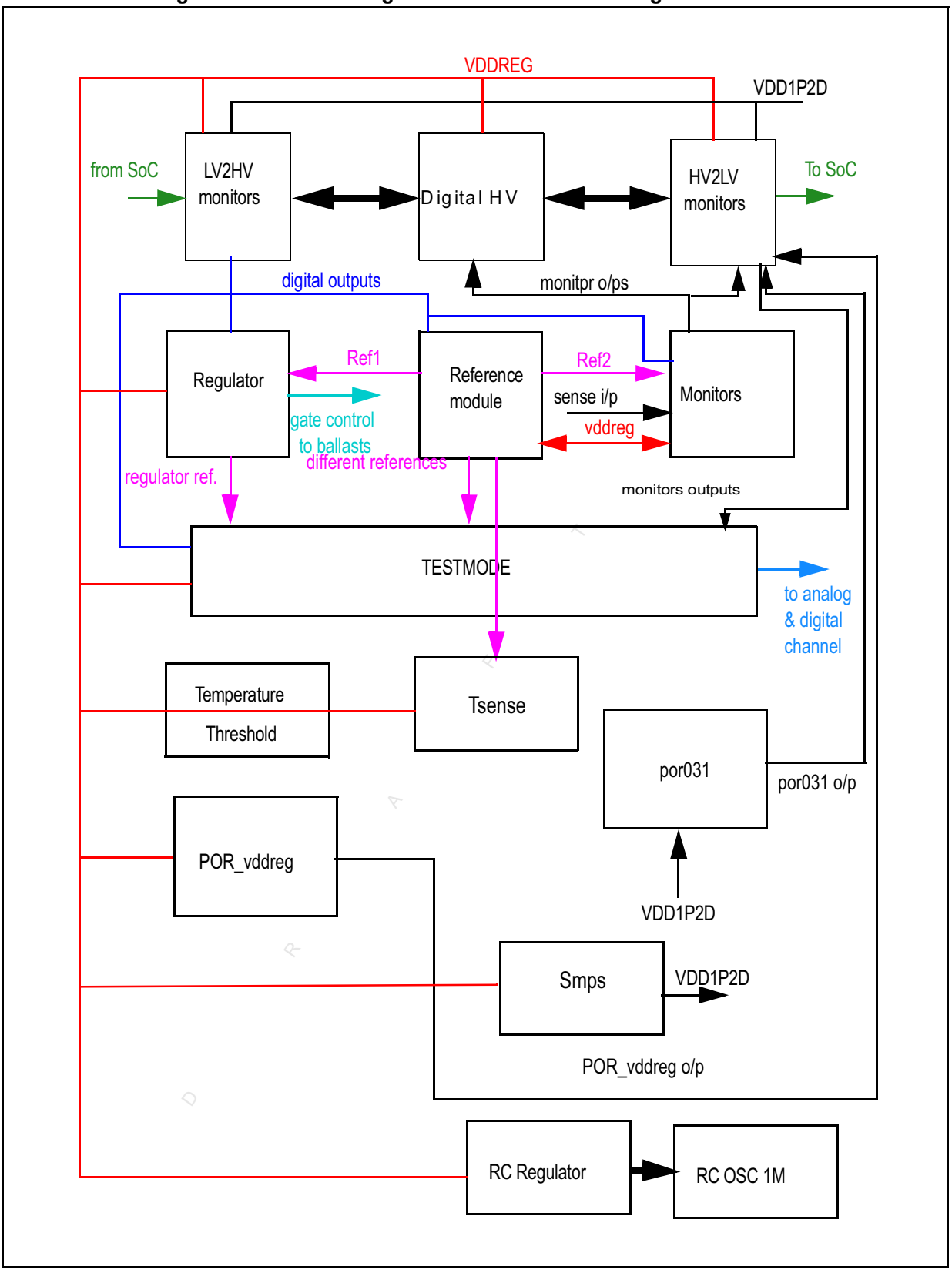
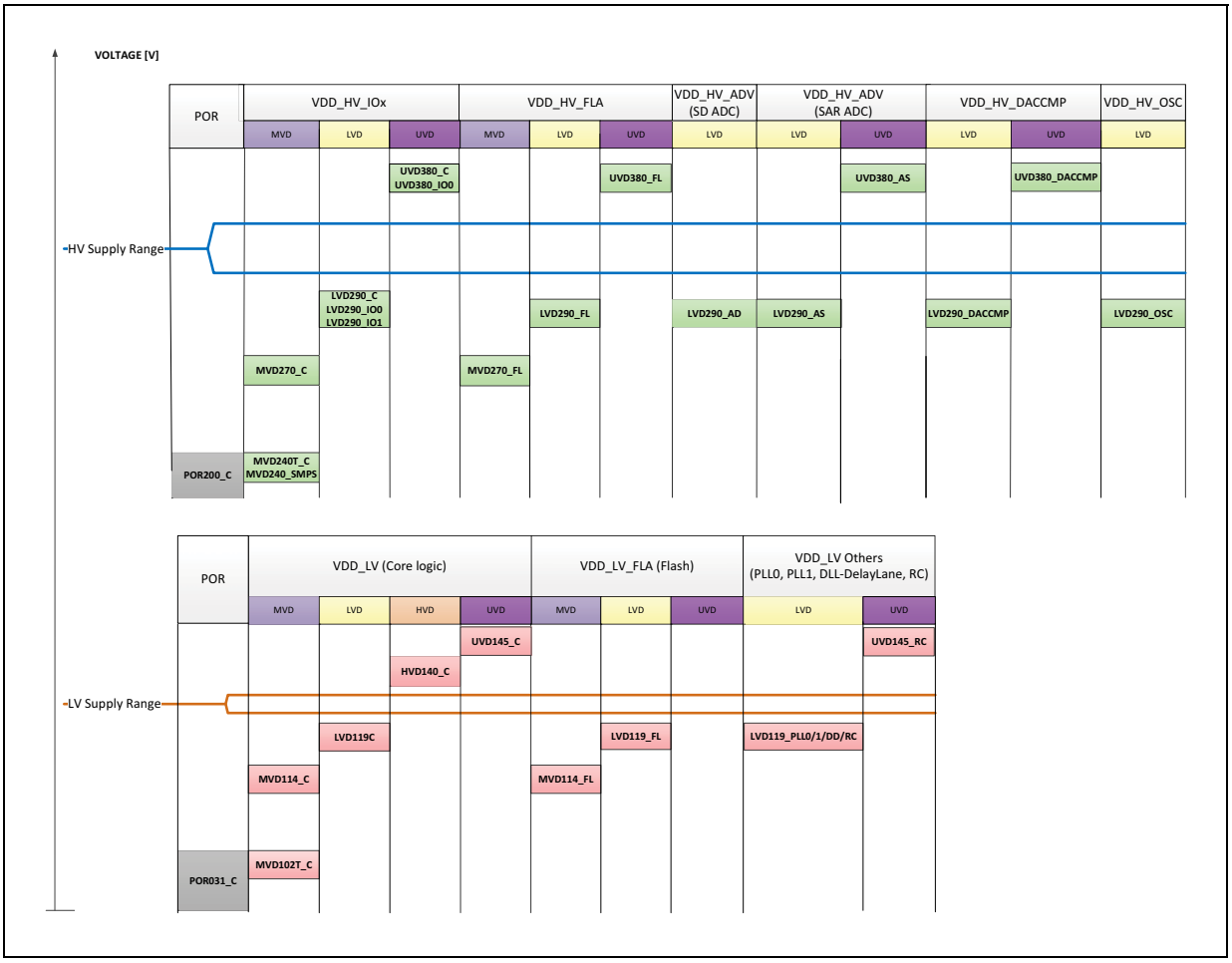


Figure 36. Voltage Monitor Coverage



Note: See Table 104: Voltage monitors description for VDs descriptions

Voltage levels for all VDs are given in the data sheet

$V_{DD_HV_IO}$ is redundant, ensuring that single pin failure does not prevent device from functioning correctly. Full specification may not be guaranteed.

V_{DD_LV} is redundant, ensuring that single pin failure does not prevent the device from functioning correctly.

8.2 Flash power requirements

During start-up operation, the flash module operates from an untrimmed regulator, which has a slightly larger variation than the ideal case. During this start-up phase, the flash can still be read, but it must be read at a slower rate. While operating in this slower read mode, the flash can operate at a much reduced voltage.

Following completion of the start-up phase the regulator is in its trimmed condition, and subsequently the flash module reference current has also been trimmed. Therefore when entering normal operation the flash can be read in the normal access manner and support the full flash specification.

8.3 Device trimming

During the initialization phase the device defaults to a pre-determined state for each of the LVDs, HVDs and the internal regulators. As the flash becomes available the differential read process allows the trimmed data to be available for trimming the internal LVDs, HVDs and regulators.

Please refer to *Section 8.5: Power sequence* for further details.

8.4 Supply monitoring (POR and LVDs)

The function of the POR and LVD circuits is to hold the device in reset regardless of how slow the supply voltage rise is, until the point at which the POR and LVDs are released.

The POR and LVD circuits function correctly even if the input voltage is non monotonic.

8.4.1 Power-on reset (POR)

The PMC implements two internal power-on reset circuits:

- POR031_C monitors the voltage on the Low Voltage input supply. It is monitoring the V_{DD_LV} pin. The POR031_C asserts a reset when the input supply is below the defined values.
- POR200_C monitors the voltage on the High Voltage input supply. It is monitoring the $V_{DD_HV_IO}$ pin. The POR trip point is high enough to make sure all the LVD circuits are functional.

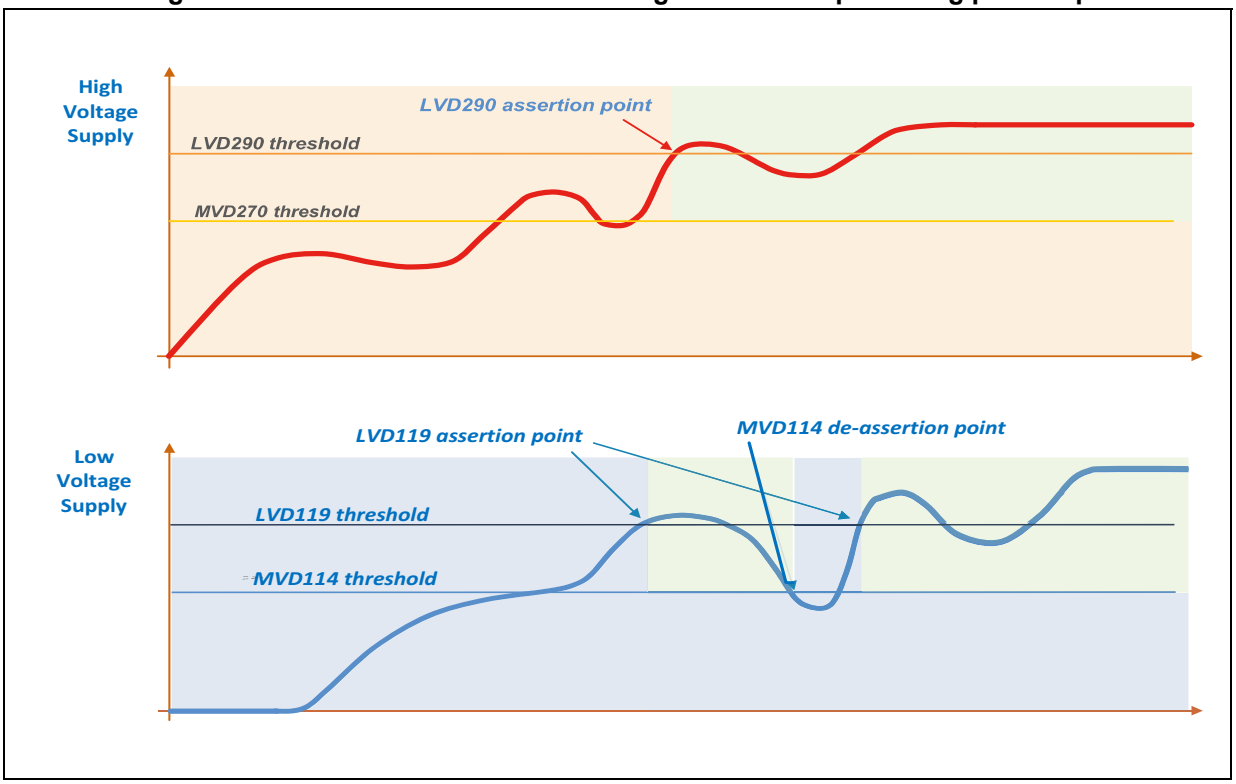
See *Chapter 6: Reset and boot* for RESETn pin functionality.

8.4.2 Behavior of device LVD / HVD

Although there is an option to disable the LVD and HVD following reset (see *Section 8.4.3: Voltage detections (MVDs, LVDs, HVDs, UVDs)*), they are capable of being used in a ‘monitor’ only mode and also capable of generating a safe / interrupt event. The diagram in the following figure illustrates the behavior of these internal LVD / HVD circuits during the power on phase, through into device initialization and subsequently in ‘normal’ run / operating mode.

The LVDs/HVDs can also be configured after device initialization preventing reset to happen when supply crosses the LVD threshold, effectively providing a higher voltage operating range. It is the responsibility of the application to ensure that the device remains in the functional range.

Figure 37. Internal LVD and HVD in configuration example during power-up



8.4.3 Voltage detections (MVDs, LVDs, HVDs, UVDs)

The internal LVD circuits monitor when the voltage on the corresponding supply is below the defined values and either assert a reset or an interrupt. The LVDs also support hysteresis in the falling and rising trip points.

8.4.3.1 LVD and HVD implementation

- All LVDs and HVDs are capable of generating either an RCC or FCCU event (or both).
- All LVDs and HVD configured for reset generation can cause functional or destructive reset. RCC PHASE0 is not exited until all destructive reset conditions are cleared.
- The appropriate bits in the PMC registers are set by LVD and HVD events. Please refer to Power management controller digital interface chapter (PMC_Dig).
- The LVDs and HVDs control is protected by the register protection scheme. Therefore it is configurable as long as the scheme is followed. Please refer to *Section 5.8.6: REG_PROT configuration*.
- SR5E1x implements REE (Reset event enable) DCF records to allow to associate “configurable” LVDs/HVDs to a RESET event. This is a write mechanism managed by SSCM during device initialization. When RESET event is selected through REE DCF record, it cannot be changed until next RESET event. When RESET event is not selected through REE DCF record, LVDs/HVDs trigger event default to FCCU event. Please refer to *Chapter 7: Device configuration format (DCF) records*.

Note: The full list of voltage monitors is provided in Table 104: Voltage monitors description. The functionality and configurability are reported in Figure 38: Voltage monitor functionality and Table 105: Voltage monitors configurability.

- When the LVD or the HVD is enabled for destructive reset generation and a subsequent trigger event is detected, the external RESETn pin is driven low.

Note: MVDs/UVDs are not configurable

Table 104. Voltage monitors description

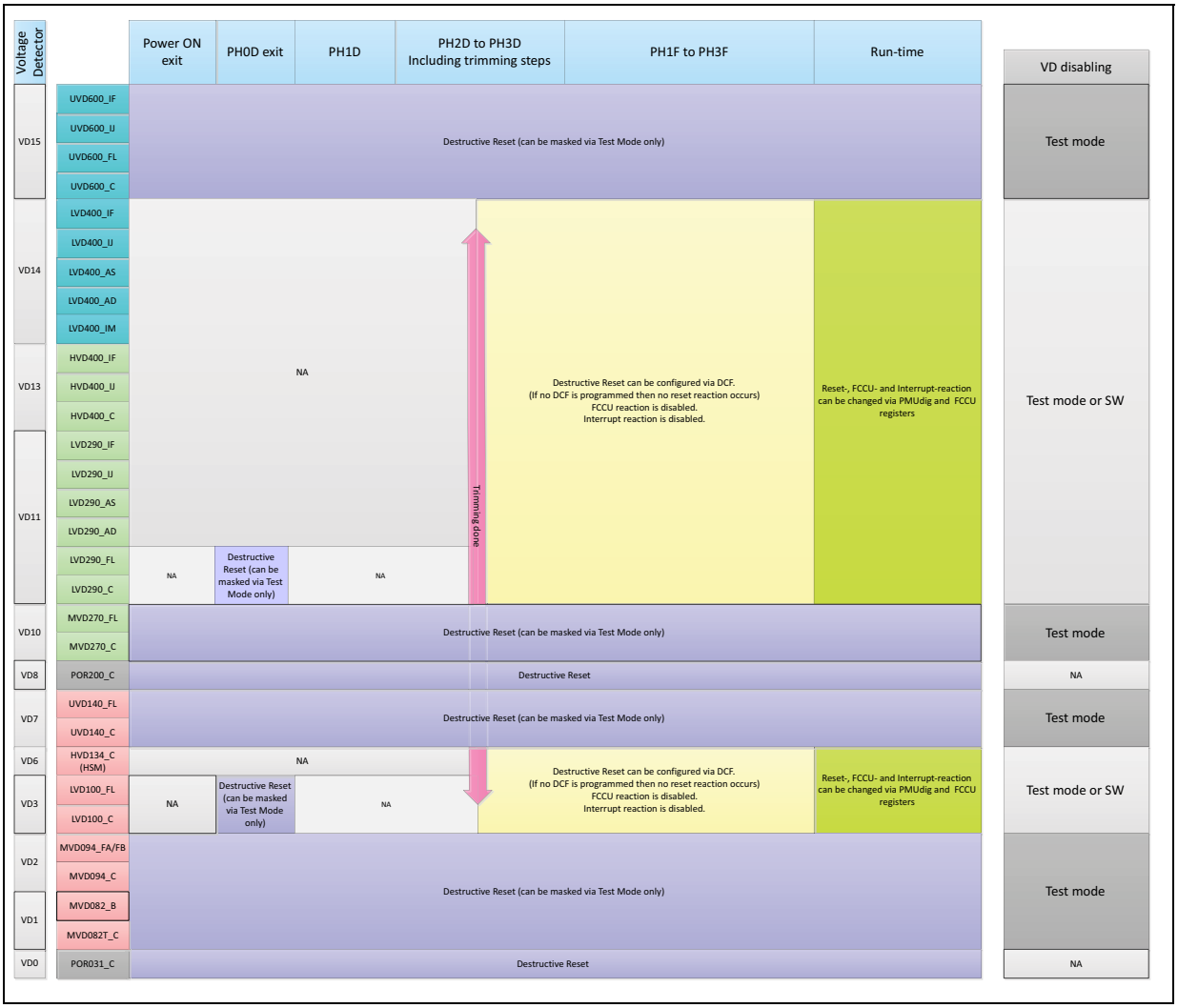
Monitor name	Voltage reference number	Description	Monitor sensing
POWER MANAGEMENT UNIT VOLTAGE MONITORS: LOW VOLTAGE SUPPLIES			
POR031_C	VD0	Low voltage supply power-on reset voltage monitor	
MVD102T_C	VD1	LV supply core minimum voltage detector	V_{DD_LV}
MVD114_C	VD2	LV supply core low range minimum voltage detector	V_{DD_LV}
MVD114_FA	VD2	LV supply flash minimum voltage detector	$V_{DD_LV_FLA}$
LVD119_C	VD3	LV supply core low voltage detector	V_{DD_LV}
LVD119_FL	VD3	LV supply flash low voltage detector	$V_{DD_LV_FLA}$
LVD119_PLL0	VD3	LV supply PLL0 low voltage detector	$V_{DD_LV_PLL0}$
LVD119_PLL1	VD3	LV supply PLL1 low voltage detector	$V_{DD_LV_PLL1}$
LVD119_DD	VD3	LV supply DLL & DelayLanes low voltage detector	$V_{DD_LV_DD}$
LVD119_RC	VD3	LV supply RCOSC low voltage detector	V_{DD_LV}
HVD140_C	VD6	LV supply core high voltage detector	V_{DD_LV}
UVD145_C	VD7	LV supply core upper voltage detector	V_{DD_LV}
UVD145_RC	VD7	LV supply RCOSC upper voltage detector	V_{DD_LV}
POWER MANAGEMENT UNIT VOLTAGE MONITORS: HIGH VOLTAGE SUPPLIES			
POR200_C	VD8	High voltage supply power-on reset voltage monitor	
MVD240T_C	VD8	HV supply core minimum voltage monitor	$V_{DD_HV_PMU}$
MVD240_SMPS	VD8	HV supply core minimum voltage monitor	$V_{DD_HV_PMU}$
MVD270_C	VD10	HV supply core minimum voltage monitor	$V_{DD_HV_PMU}$
MVD270_FL	VD10	HV supply flash minimum voltage monitor	$V_{DD_HV_FLA}$
LVD290_C	VD11	HV supply core low voltage monitor	$V_{DD_HV_PMU}$
LVD290_FL	VD11	HV supply flash low voltage monitor	$V_{DD_HV_FLA}$
LVD290_AD	VD11	HV supply SD-ADC low voltage monitor	$V_{DD_HV_SD}$
LVD290_AS	VD11	HV supply SAR-ADC low voltage monitor	$V_{DD_HV_SAR}$
LVD290_DACCMP	VD11	HV supply DAC & COMP low voltage monitor	$V_{DD_HV_DAC}$

Table 104. Voltage monitors description (continued)

Monitor name	Voltage reference number	Description	Monitor sensing
LVD290_OSC	VD11	HV supply OSC low voltage monitor	$V_{DD_HV_OSC}$
LVD290_IO1	VD11	HV supply I/O low voltage monitor	$V_{DD_HV_IO1}$ segment
LVD290_IO0	VD11	HV supply I/O low voltage monitor	$V_{DD_HV_IO0}$ segment
UVD380_C	VD15	HV supply core upper voltage monitor	$V_{DD_HV_PMU}$
UVD380_FL	VD15	HV supply flash upper voltage monitor	$V_{DD_HV_FLA}$
UVD380_IO0	VD15	HV supply I/O upper voltage monitor	$V_{DD_HV_IO0}$ segment
UVD380_AS	VD15	HV supply SAR-ADC upper voltage monitor	$V_{DD_HV_SAR}$
UVD380_DACCMP	VD15	HV supply DAC & COMP upper voltage monitor	$V_{DD_HV_DAC}$

The following figure provides the functionality of monitors depending on configuration.

Figure 38. Voltage monitor functionality



The following table provides the monitor status depending on configuration.

Table 105. Voltage monitors configurability

Monitor type	Reset Event Enable(1)	Reset Event Select(1)	Event Pending Register(1)	Interrupt Enable(1)	FCCU Event Enable(1)	Reset Event Enable DCF(1)
MVDs	No	No	No	No	No	No
LVDs	Yes	Yes	Yes	Yes	Yes	Yes
HVDs	Yes	Yes	Yes	Yes	Yes	Yes
UVDs	No	No	No	No	No	No

1. Refer to the Power management controller digital interface (PMC_Dig) chapter.

8.5 Power sequence

The following sections describe the power sequence and the relation among the different supplies during power-up and power-down.

The device is considered to be in a power sequence (or POWERUP state) when it is either not supplied or partially supplied. An internal power-on signal is used to identify POWERUP state. This signal is released high on exit of power sequence. The power-on signal is a combination of the LVDs monitoring the following supplies:

- V_{DD_LV}
- $V_{DD_HV_IO}$
- $V_{DD_HV_FLA}$

The actual threshold used for each LVD is dependent on the configuration of the device. This is configurable by hardware (flash option bits content) or by software (LVD event configuration through register interface). Once power-on signal has been asserted, device configuration is reset to default power-up configuration.

8.5.1 Power-up sequence

In this section it is assumed that all supplies are low when entering the power-up sequence. Brown-out and power down sequences are managed in specific sections.

8.5.1.1 POWERUP state

At the beginning of power-up, the internal power-on signal remains low due to internal parasitics diodes. As soon as minimum threshold is reached on V_{DD_LV} and V_{DD_HV} supply, the internal power-on signal is forced low until power-up LVDs reach upper not trimmed threshold (refer to datasheet).

The different supplies can rise independently as long as the constraints described in the device datasheet are met.

During power-up, all functional terminals are maintained into a known state as described in the device datasheet.

8.5.1.2 POWERUP exit

The POWERUP state can be exited when both V_{DD_LV} and V_{DD_HV} are above the internal LVD thresholds.

8.5.1.2.1 V_{DD_LV} conditions and LVD trimming/enabling sequence

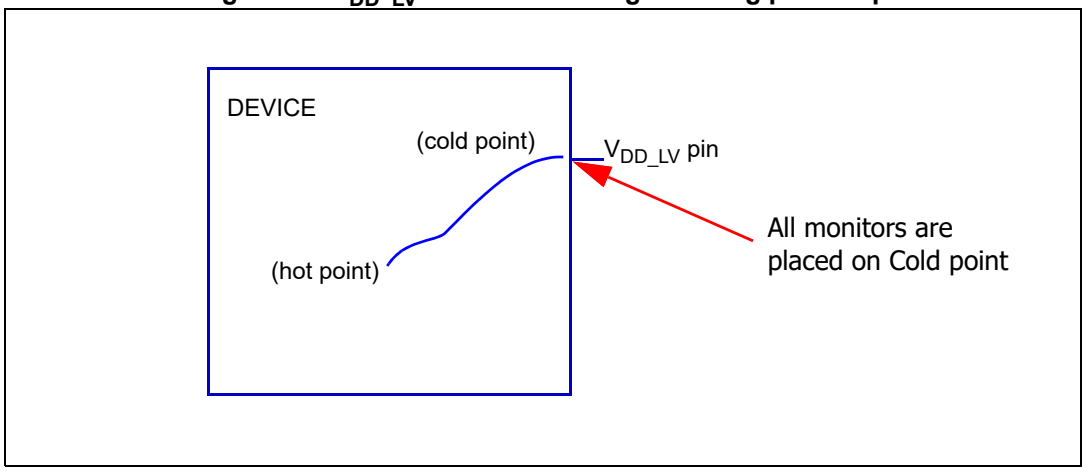
During POWERUP, only the following low voltage VDs are enabled:

- MVD102T_C, MVD114_C, MVD114_FL, UVD145_C,

The V_{DD_LV} conditions to exit POWERUP are the following:

- MVD114_C, MVD114_FL upper threshold is crossed

Figure 39. V_{DD_LV} monitored voltages during power-up



After the POWERUP exit conditions (low voltage, high voltage conditions) have been verified, the internal power-on signal is released to all analog modules.

The internal RC oscillator module (IRCOSC) starts initialization and provides clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles and LVD119_C is masked. Only MVD114_C remains active.

MVD114_C and LVD119_C share the same reference. This difference provides margin with respect to maximum internal resistive drop and hysteresis addressing external supply.

The device proceeds through the reset sequence through RCC phase PHASE0, PHASE1[DEST], PHASE2[DEST] and PHASE3[DEST].

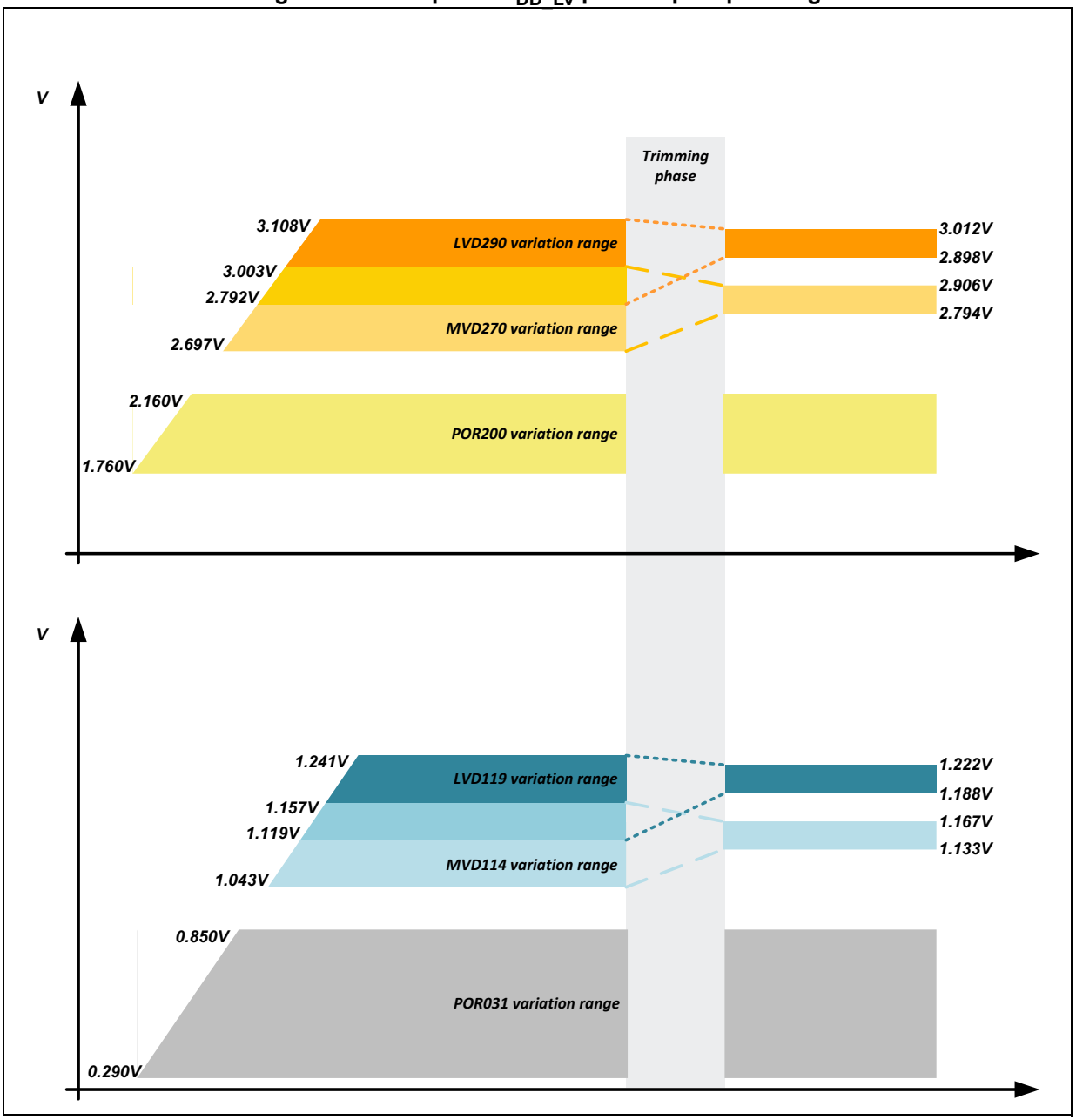
Voltage detector (LVD/HVD) modules are trimmed at the beginning of PHASE3[DEST]. Trimming of LVDs/HVDs are stored as internal DCF records. They are read by SSCM at low voltage, using the differential flash read accesses and applied to each LVDs/HVDs module. After analog delay ($t_{LVDTRIM}$) has elapsed, the PMC acknowledges that all LVDs/HVDs have been trimmed and supply is above threshold, the SSCM proceeds with the reset sequence, eventually running full speed accesses to the flash to read the option bits required to complete device configuration.

The configurable LVD/HVD modules can optionally be unmasked at the end of PHASE3[DEST]. Mask information is read as DCF record from the flash UTEST option bits.

When LVD119_C and LVD290_C are masked by the application using the flash user option bits, the device relies on RESETn signal to detect a voltage failure during power-up. The device must wait for RESETn to be released high before proceeding with the power-up sequence.This may increase the amount of time necessary to complete the reset sequence.

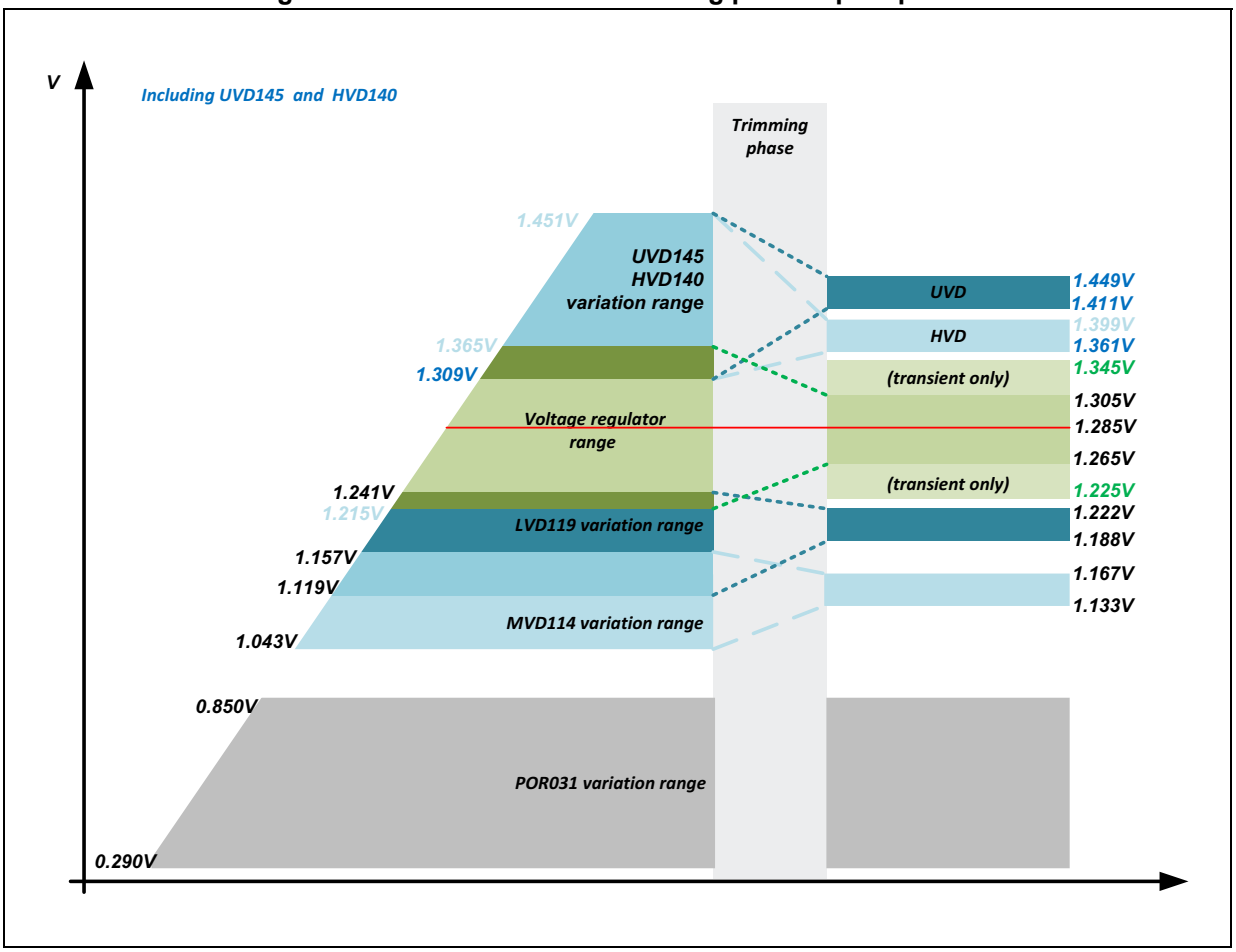
The following figure provides an example of V_{DD_LV} and V_{DD_HV} power-up sequence before and after the monitor trimming.

Figure 40. Example of V_{DD_LV} power-up sequencing



The following figure provides an example of V_{DD_LV} power-up sequence before and after the monitor trimming including UVD and HVD monitors.

Figure 41. Threshold variation during power-up sequence



8.5.1.2.2 V_{DD_HV} conditions and LVD trimming/enabling sequence

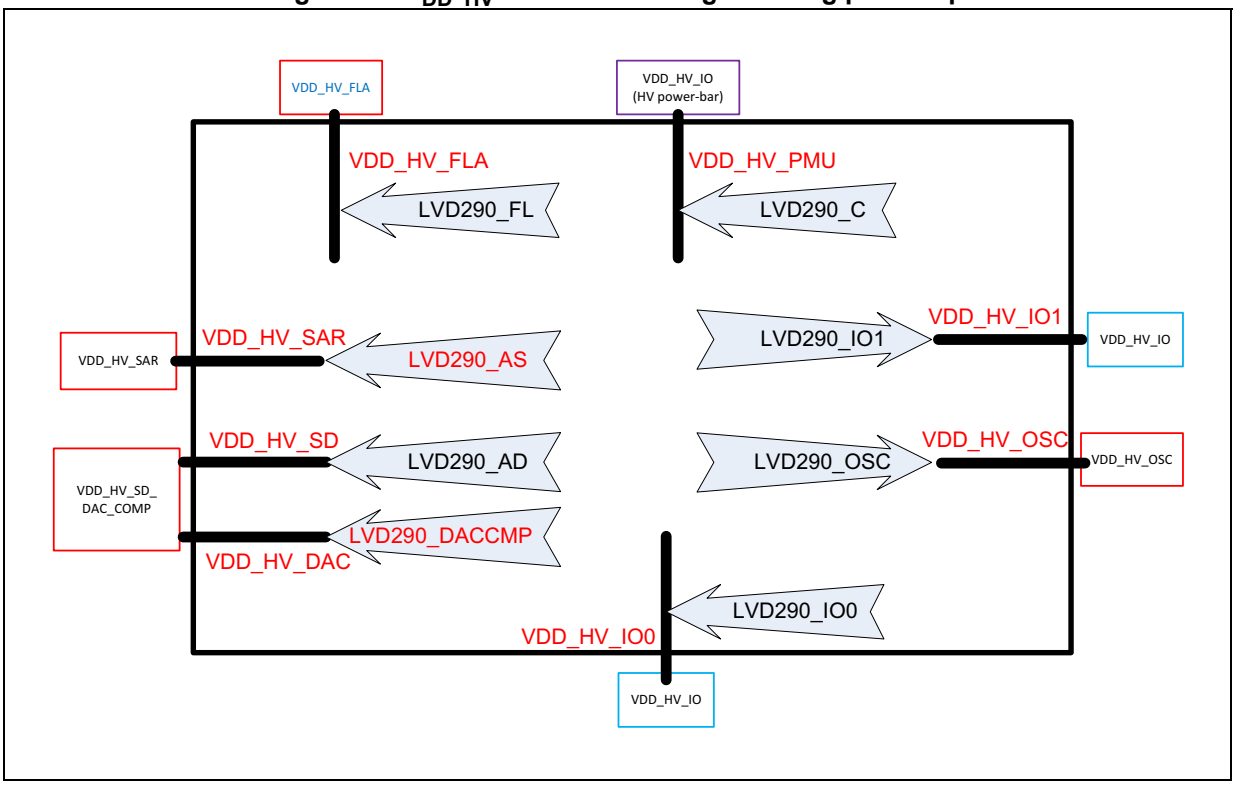
During POWERUP, the following high voltage MVDs are enabled:

- MVD270_C
- MVD270_FL

The V_{DD_HV} conditions to exit POWERUP are the following:

- LVD290_C
- LVD290_FL

Figure 42. V_{DD_HV} monitored voltages during power-up



The IRCOSC module starts initialization and provides the clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles.

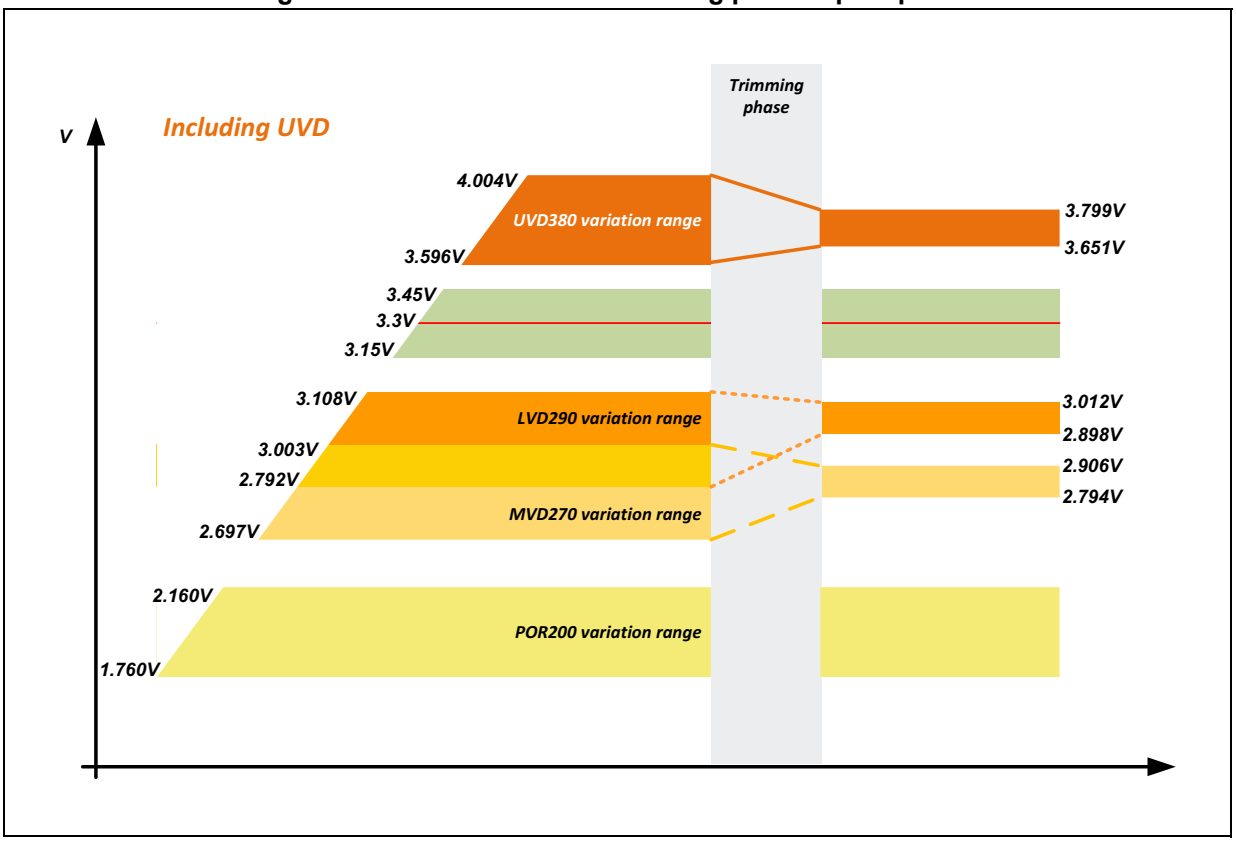
Device proceeds through the reset sequence through RCC phase PHASE0, PHASE1[DEST], PHASE2[DEST], PHASE3[DEST].

LVD/HVD modules are trimmed at the beginning of PHASE3[DEST]. Trimming is done by SSCM at low voltage, using the differential flash read accesses. After trimming is completed, SSCM waits for PMC acknowledgement before proceeding with reset sequence.

Configurable LVD/HVD modules are optionally enabled at the end of PHASE3[DEST].

The following figure provides the threshold variation of V_{DD_HV} LVDs monitor during power-up sequence from POWERUP before and after trimming.

Figure 43. Threshold variation during power-up sequence



The device needs to fulfill all conditions described in this section to recover.

8.5.2 Power-down sequence

In case LVDs/HVDs are configured to generate destructive reset, either LVD is below threshold or HVD is above threshold, the device enters PHASE0 phase.

Power-down sequence is actually entered as soon as the threshold of MVD114 and MVD270 is crossed. The device enters the POWERUP state. The internal power-on signal is asserted.

On power-on signal assertion, all modules reach their safe state.

Device supplies may then proceed to drop down to ground either through device leakage or external pull-down. During power-down, all supplies comply with supply constraints.

8.5.3 Brown-out management

During brown-out, the device re-enters the POWERUP phase as soon as the threshold of either MVD114 or MVD270 is crossed. The internal power-on signal is asserted.

Note: The device correctly starts independently from any residual voltage. Below MVD270 *threshold the device is under POWERUP state. Above the* MVD270 *threshold, the device is into a RESET state as defined by Voltage monitor or external RESETn pin or active state when all have been released.*

On internal power-on signal assertion, all analog modules reach their safe state.

8.5.4 Low voltage requirement during crank

The device can continue operation to the minimum input voltage during crank.

In order to proceed with execution during cranking and prevent device reset, it is important to correctly configure the high voltage LVDs.

During device switch-off the external LVDs monitoring may not be activated quickly enough (external assertion of the RESETn pin) and hence the –10% level is crossed before POR is asserted. Malfunction is possible in this case, but reliability is not impacted. In particular the flash is not corrupted.

Internal LVDs that monitor the supply ensure that the flash content is protected.

9 Security

All family devices have a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access. Some security features are available in all device configurations while other more advanced security features are not available in all device configurations. These more advanced features require more than simply being turned on or off; they require significant involvement for implementation. Consequently, they provide an important opportunity to define the security level of the device.

9.1 Basic security

The SR5E1x device has the following basic set of security features.

- Device security feature based on the life cycle model with code and data access progressively more restricted as device matures through defined life cycle steps
- Memory security features:
  - NVM censorship support, password protection, one-time-programmable (OTP) flash memory areas, Flash erase counter and tamper detection
  - SRAM and caches initialized to a constant value after reset (power-on reset, when in a test mode, when not booting from internal flash memory); MBIST functionality used in case of power-on and destructive resets to initialize RAMs selected via configuration in UTEST flash; BAF software routines to initialize RAMs when not booting from internal flash memory
- Monitoring of operating conditions
- Unique ID for each device: each SR5E1x device has a unique identification number stored in an OTP flash memory area which can be read by any of the cores on the device
- Secure watchdog timer
- Debugger access control

9.2 Advanced security

Depending on the device configuration, the following features are available:

- Customer programmable Hardware Security Module (HSM): a dedicated security subsystem that includes a processor core, dedicated SRAM, an encryption module, and exclusive access to secure areas of device flash memory; HSM runs code independently from the main device processor cores and can be used to implement advanced security and monitoring functions
- Advanced debugger access control
- Boot modes:
  - Trusted/secure boot support
  - Handshake with BAF supported

9.3 Detailed security information

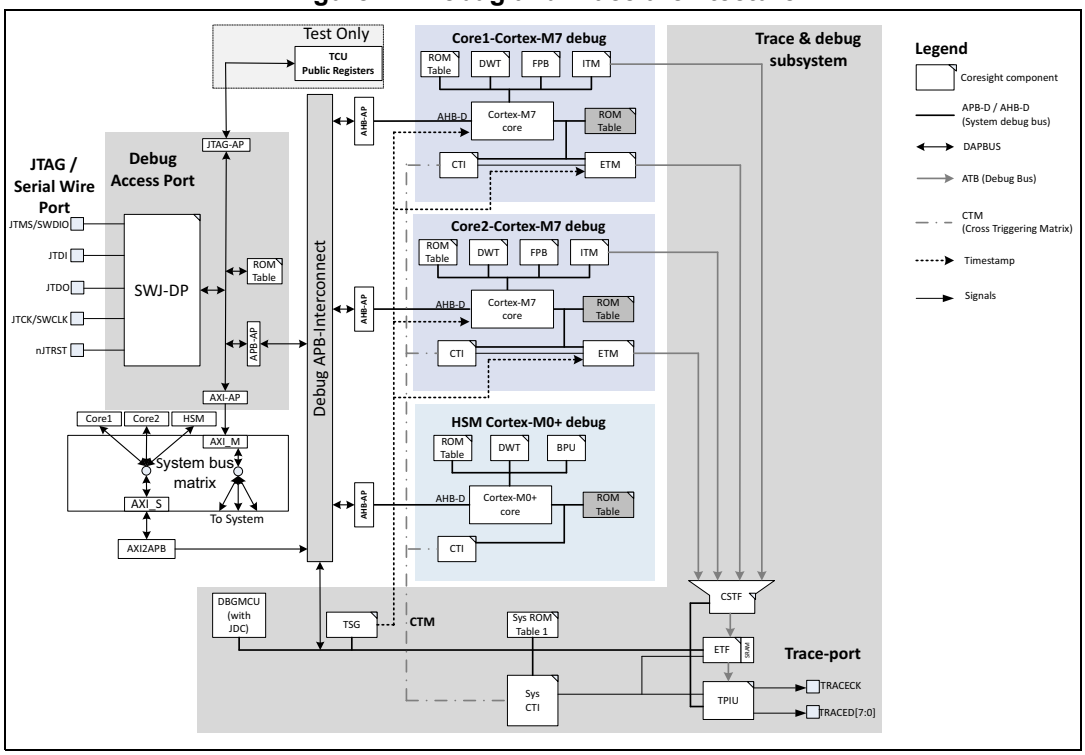
Details of most of the SR5E1x security features are published in a separate *SR5E1x Microcontroller Security Reference Manual*, which is only available to qualified customers.

10 Debug support (DBG)

10.1 Debug and trace architecture

The debug and trace architecture is shown in the figure below.

Figure 44. Debug and Trace architecture



10.2 Debug infrastructure features

A comprehensive set of trace and debug features is provided to support software development and system integration:

- Independent breakpoint debugging of each CPU core in the system
- Code execution tracing for M7 cores
- Data Tracing for M7 cores
- Software instrumentation
- Cross-triggering
- JTAG debug port
- Serial-wire debug port
- Trace port
- Arm® CoreSight™ debug and trace components

10.3 Debug access port functional description

The debug access port (DAP) is a debug subsystem comprising serial-wire and JTAG debug port (SWJ-DP) and two access ports.

10.3.1 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

The port can be configured as:

- a 4-pin standard JTAG debug port (JTAG-DP)
- a 2-pin (clock + data) “serial-wire” debug port (SW-DP)

The two modes are mutually exclusive, since they share the same IO pins.

By default, the JTAG-DP is selected upon a system or power-on reset. The five IOs are configured by hardware in debug alternative function mode.

The SWJ-DP incorporates pullup resistors on the JTDI and JTMS/SWDIO lines, as well as pull-down resistors on the JTCK/SWCLK and JCOMP lines.

10.3.2 Access ports

There are the following access ports (AP) attached to the DP:

- APB-AP: for connection to APB-Interconnect, that provides access to Cores Debug Access Port and to the all system level CoreSight™ components.
  - Three AHB-APs are present between APB debug and each AHBD port of cores. AHBD port give access to the debug and trace features integrated in the cores (which means in Cortex-M7 processor cores and in Cortex-M0+ processor).
  - A branch of APB-Interconnect allows the access to the debug and trace features on the system APB debug bus for all components not included in processor cores.
- AXI-AP: allows access of the debug to the NIC as master. This gives visibility of system memories and peripherals. No CoreSight™ components are accessible through this port.
- JTAG_AP: non user port, connected to public TCU registers, that allows to have visibility to device status, monitoring what happens during reset phases.

10.4 NIC ports related to debug

The NIC has an initiator AXI port for the access from the DAP to all device resources (memories, peripherals).

The NIC has an APB slave port for the access from Core1 and Core2 and CoreH to the APB-Debug, that allows the programming of the debug components.

10.5 Global timestamp generator (TSG)

The global timestamp generator contains a 64-bit counter that provides a common timing reference for all of the trace sources in the system, namely the ETM and ITM in each processor core.These components insert timestamps in the trace streams that allow the

trace analyzer to recover the chronological order of trace packets, which can be lost when multiple trace sources are multiplexed into one stream at the funnels.

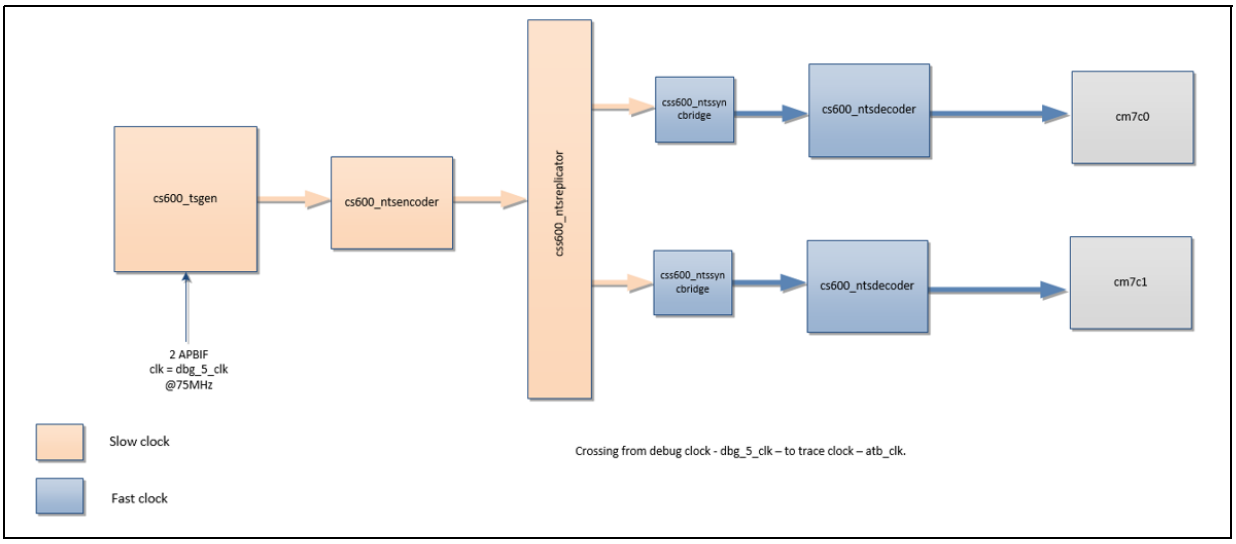
The TSG registers are accessible over the APB-D. This allows the debugger or debug software to:

- Start and stop the timestamp incrementing
- Read the current timestamp value
- Change the current timestamp value
  - The timestamp counter must be halted while it is changed. When the timestamp value is changed, the timestamp generator resynchronizes all the trace sources.
- change the reported timestamp increment

The timestamp is distributed to the all Cortex-M7 cores.

Timestamp structure is shown in the figure below.

Figure 45. Timestamp structure



10.6 Cross trigger interfaces (CTI) and matrix (CTM)

The cross trigger interfaces (CTI) and cross trigger matrix (CTM) together form the CoreSight embedded cross trigger feature. There are four CTI components, one at system level, two dedicated to each Cortex-M7 and one dedicated to the CortexM0+ inside HSM.

The four CTIs are connected to each other via the CTM.

The system-level CTI is accessible to the debugger via the system access port and associated APB-D.

The Cortex- M7 CTI is physically integrated in the Cortex-M7 core, and is accessible via the Cortex-M7 access port and associated AHBD.

The Cortex- M0+ CTI is physically integrated in the Cortex-M0+ core, and is accessible via the Cortex-M0+ access port and associated AHBD.

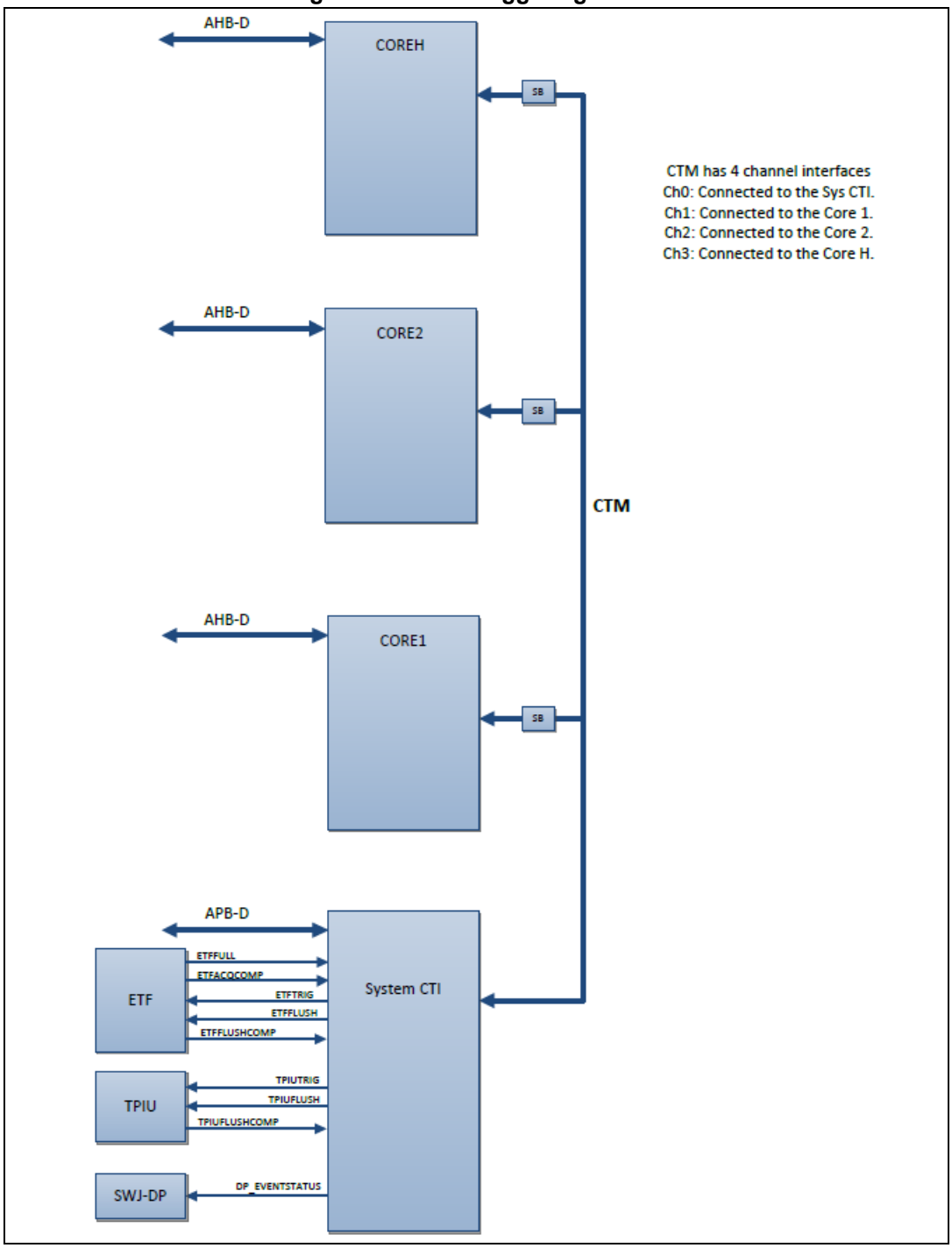
Two internal trigger sources are managed by CTM network: Halt and Restart.

No external triggers have to be managed.

In addition, TPIU and ETF triggers are connected to the system CTI. For more details about the triggers, refer to Arm® CoreSight™ System-on-Chip SoC-600 revision r4p0. Furthermore, the dp_eventstatus has been used: a generic trigger can be mapped on the dp_eventstatus to quickly inform the external tool through the DP when the particular trigger is received.

The figure below shows the cross triggering matrix.

Figure 46. Cross triggering matrix



10.6.1 Embedded trace FIFO (ETF)

The ETF is a 4 Kbyte memory that captures trace data from four trace sources, namely the ETM and ITM of each M7- CPU core. The ETF is a design configuration of the CoreSight™ trace memory controller component.

The ETF can be used in three modes (selected in the mode register):

1. Hardware FIFO mode
2. Software FIFO mode
3. Circular buffer mode

10.7 Trace port interface unit (TPIU)

The TPIU is a CoreSight™ component that formats the trace stream and outputs it on the external trace port signals. The TPIU has a single ATB slave port for incoming trace data.

The trace port is a synchronous parallel port, comprising a clock output, TRACECK, and up to eight data outputs, TRACED(7:0). The trace port width is programmable in the range 1 to 8.

Using a smaller port width reduces the number of test points/connector pins needed, and frees up IOs for other purposes. However it restricts the bandwidth of the trace port and hence the quantity of trace information that can be output in real time.

TRACECK frequency is programmable and max frequency is 50MHz.

10.8 Cortex-M7 debug functional description

The Cortex-M7 subsystem features the following CoreSight™ components:

- ROM tables
- System control space (SCS)
- Breakpoint unit (FPB)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded trace macrocell (ETM)
- Cross trigger interface (CTI)

These components are accessible by the debugger via the Cortex-M7 AHB-AP and its associated AHBD bus.

ETM has to be configured in order to trace both instruction and data.

10.9 Debug configuration in lockstep

When lockstep is enabled, only Core1 is accessible for debug.

The port AHB-AP of Core2 is closed by gating with lockstep information.

When lockstep feature is enabled, the user must ensure the correspondent ATB Funnel slave interfaces are disabled to avoid funnel switching activity and trace bandwidth

reducing. Therefore, when lockstep is enabled, FUNNELCONTROL[7:3] must be equal to 0x0.

Table 106. Slave mapping

Funnel control [bits]	Slave interface
[0]	Core1_ITM_Trace_Data
[1]	Core1_ETM_Trace_Instruction
[2]	Core1_ETM_Trace_Data
[3]	Core2_ITM_Trace_Data
[4]	Core2_ETM_Trace_Instruction
[5]	Core2_ETM_Trace_Data

Note: For information on the CoreSight™ registers, refer to the documentation available on the arm website.

10.10 Additional requirements

10.10.1 Capability of the debugger host to connect under system reset

The Cortex®-M7 differentiates the reset of the debug part (generally PORRESETn) and the other one (SYSRESETn).

This way, it is possible for the debugger to connect under system reset, programming the core debug registers to halt the core when fetching the reset vector.

Then the host must release the system reset and the core immediately halts without having executed any instructions.

Note: The debugger host can always connect under system reset with the Boot CPU (Core1) while for Core2 the enable of core is mandatory by means of DCF Global System Configuration[CORE2_EN].

In addition, it must be possible to program any debug features under system reset.

Note: The possibility to program debug features under reset has some limitations in SR5E1x, agreed with Design Team and listed below like:

- *Trace components (TPIU, EFT, Funnel) cannot be programmed under reset*
- *Programming is possible only from external tool, not from cores*

10.10.2 Capability to control debug features of cores and IPs

The capability to control debug features of cores and IPs includes the possibility to:

- Maintain the clock to the processor cores when in low-power modes (C-sleep mode).
- Maintain the clock to the system debug and trace components when in low power modes.
- Stop the clock to certain peripherals (CAN, SMBUS timeout, Watchdogs, Timers, RTC) when one of the processor cores is stopped in debug mode.
- For timers having complementary outputs (TIM1/8/15/16), the outputs must be disabled (as if the MOE bit was reset) for safety purposes when the counter is stopped (TIM1/8/15/16).

These functionalities are implemented inside DBGMCU.

10.10.3 Debug protection

Debug access has to be controlled by life cycle mechanism.

For details, refer to Stellar E security reference manual.

10.11 DBGMCU module

DBGMCU allows to control the behavior of the peripherals during a breakpoint.

During a breakpoint, it is necessary to choose how the counter of timers, RTC and watchdog must behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the I2C, the user can choose to block the SMBUS timeout during a breakpoint.

See also requirements in *Section 10.10.2: Capability to control debug features of cores and IPs*.

The DBGMCU freeze registers can be written by the debugger under system reset.

Two sets of freeze registers must be implemented, one for each core, for all the peripherals that support freeze feature.

When the FRZ bit related to a peripheral is set, the peripheral is frozen when the corresponding core (Core1 or Core2) is halted in debug mode.

For the mapping of freeze registers, refer to *Section 10.13: DBGMCU register descriptions*.

10.12 DBGMCU register summary

Table 107. DBGMCU register list

Offset	Register Name
0x0000	*APB1_C1_FZ_REG*
0x0004	*APB2_C1_FZ_REG*

Table 107. DBGMCU register list (continued)

Offset	Register Name
0x0008	*AXI_C1_FZ_REG*
0x0010	*APB1_C2_FZ_REG*
0x0014	*APB2_C2_FZ_REG*
0x0018	*AXI_C2_FZ_REG*
0x0020	*DBG_RST_REG*
0x0024	*DOUT_REG*
0x0028	*DIN_REG*
0x002C	*DMCR*
0x0030	*DMSR*
0x0040	*CTI_TRG_SEL*
0x0044	*DBG_SLEEP_CFG*
0x0FBC	*DEVARCH*
0x0FCC	*DEVTYPE*
0x0FD0	*PIDR4*
0x0FD4	*PIDR5*
0x0FD8	*PIDR6*
0x0FDC	*PIDR7*
0x0FE0	*PIDR0*
0x0FE4	*PIDR1*
0x0FE8	*PIDR2*
0x0FEC	*PIDR3*
0x0FF0	*CIDR0*
0x0FF4	*CIDR1*
0x0FF8	*CIDR2*
0x0FFC	*CIDR3*

10.13 DBGMCU register descriptions

APB1_C1_FZ_REG APB1 Debug Freeze Register Core 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCCU_DBG	RESERVED4								I2C2_FRZ	I2C1_FRZ	RESERVED3	TIM_TS_FRZ	TIM7_FRZ	TIM6_FRZ	TIM3_FRZ	TIM2_FRZ	RESERVED2				RTC_FRZ	RESERVED1				IWDG2_FRZ	IWDG1_FRZ	RESERVED0		WWDG2_FRZ	WWDG1_FRZ
RW	R								RW	RW	R	RW	RW	RW	RW	RW	R				RW	R				RW	RW	R		RW	RW

Address: DBGMCUBaseAddress + 0x0000

Type: RW

Reset: 0x0000

Description: This register is used to enable the debug freeze feature of APB1 memory mapped IPs for the core 1. This register can be written and read only if DBGEN is active - DBGMCU invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31] **FCCU_DBG**: **“**FCCU Debug Mode”
0x0: “FCCU Normal Operation”
0x1: “FCCU in debug mode when core 1 is halted. When FCCU_CTRL[DEBUG] is asserted and this bit is set, FCCU moves into debug state and suspends the transition of the FSM by suspending timer elapse time effect.”

[30:23] **RESERVED4**: “This field is reserved. It must be kept to zero by the software.**”**

[22] **I2C2_FRZ**: “I2C 2 SMBUS Timeout Freeze”
0x0: “I2C 2 SMBUS Timeout is not blocked when Core 1 is halted.”
0x1: “I2C 2 SMBUS Timeout is blocked when Core 1 is halted.”

[21] **I2C1_FRZ**: **“**I2C 1 SMBUS Timeout Freeze**”**
0x0: “I2C 1 SMBUS Timeout is not blocked when Core 1 is halted.”
0x1: “I2C 1 SMBUS Timeout is blocked when Core 1 is halted.”

[20] **RESERVED3**: **“**This field is reserved. It must be kept to zero by the software.”

[19] **TIM_TS_FRZ**: “Timer TS Debug Freeze”
0x0: “Timer TS is not disabled when Core 1 is halted.”
0x1: “Timer TS is disabled when Core 1 is halted.”

[18] **TIM7_FRZ**: **“**Timer 7 Debug Freeze**”**
0x0: “Timer 7 is not disabled when Core 1 is halted.”
0x1: “Timer 7 is disabled when Core 1 is halted.”

[17] **TIM6_FRZ**: **“**Timer 6 Debug Freeze**”**
0x0: “Timer 6 is not disabled when Core 1 is halted.”
0x1: “Timer 6 is disabled when Core 1 is halted.”

[16] **TIM3_FRZ**: "Timer 3 Debug Freeze"
0x0: "Timer 3 is not disabled when Core 1 is halted."
0x1: "Timer 3 is disabled when Core 1 is halted."

[15] **TIM2_FRZ**: "Timer 2 Debug Freeze"
0x0: "Timer 2 is not disabled when Core 1 is halted."
0x1: "Timer 2 is disabled when Core 1 is halted."

[14:11] **RESERVED2**: "This field is reserved. It must be kept to zero by the software."

[10] **RTC_FRZ**: "RTC Debug Freeze"
0x0: "RTC is not disabled when Core 1 is halted."
0x1: "RTC is disabled when Core 1 is halted."

[9:6] **RESERVED1**: "This field is reserved. It must be kept to zero by the software."

[5] **IWDG2_FRZ**: "Independent WDG 2 Debug Freeze"
0x0: "Independent watchdog 2 is not disabled when Core 1 is halted."
0x1: "Independent watchdog 2 is disabled when Core 1 is halted."

[4] **IWDG1_FRZ**: "Independent WDG 1 Debug Freeze"
0x0: "Independent watchdog 1 is not disabled when Core 1 is halted."
0x1: "Independent watchdog 1 is disabled when Core 1 is halted."

[3:2] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[1] **WWDG2_FRZ**: "Window WDG 2 Debug Freeze"
0x0: "Window watchdog 2 is not disabled when Core 1 is halted."
0x1: "Window watchdog 2 is disabled when Core 1 is halted."

[0] **WWDG1_FRZ**: "Window WDG 1 Debug Freeze"
0x0: "Window watchdog 1 is not disabled when Core 1 is halted."
0x1: "Window watchdog 1 is disabled when Core 1 is halted."

APB2_C1_FZ_REG APB2 Debug Freeze Register Core 1

31	30	29	28	27 26	25	24	23 22 21 20 19	18	17	16	15	14 13	12	11	10 9 8 7 6 5 4	3	2	1	0
FDCAN4_DIS_MORD	FDCAN3_DIS_MORD	FDCAN2_DIS_MORD	FDCAN1_DIS_MORD	RESERVED3	SDADC2_DBG_FRZ	SDADC1_DBG_FRZ	RESERVED2	TIM16_PWM_FRZ	TIM15_PWM_FRZ	TIM5_PWM_FRZ	TIM4_PWM_FRZ	RESERVED1	TIM8_PWM_FRZ	TIM1_PWM_FRZ	RESERVED0	TSU_CAN4_DIS_MORD	TSU_CAN3_DIS_MORD	TSU_CAN2_DIS_MORD	TSU_CAN1_DIS_MORD
RW	RW	RW	RW	R	RW	RW	R	RW	RW	RW	RW	R	RW	RW	R	RW	RW	RW	RW

Address: DBGMCUBaseAddress + 0x0004

Type: RW

Reset: 0x0000

Description: This register is used to enable the debug freeze feature of APB2 memory mapped IPs for the core 1. This register can be written and read only if DBGEN is active - DBGMCU invasive debug.
access_size: 32

min_write_access_size: 32
min_read_access_size: 32

[31] **FDCAN4_DIS_MORD**: "FDCAN 4 Disable Modify Read"
0x0: "See FDCAN 1 Disable Modify Read."
0x1: "See FDCAN 1 Disable Modify Read."

[30] **FDCAN3_DIS_MORD**: "FDCAN 3 Disable Modify Read"
0x0: "See FDCAN 1 Disable Modify Read."
0x1: "See FDCAN 1 Disable Modify Read."

[29] **FDCAN2_DIS_MORD**: "FDCAN 2 Disable Modify Read"
0x0: "See FDCAN 1 Disable Modify Read."
0x1: "See FDCAN 1 Disable Modify Read."

[28] **FDCAN1_DIS_MORD**: "FDCAN 1 Disable Modify Read"
0x0: "FDCAN modify read is not disabled when Core 1 is halted."
0x1: "FDCAN modify read is disabled when Core 1 is halted: it means that the register bit fields that change after a read operation in normal mode are frozen in debug mode. Therefore a read operation of this register does not change the content."

[27:26] **RESERVED3**: "This field is reserved. It must be kept to zero by the software."

[25] **SDADC2_DBG_FRZ**: "SDADC 2 Debug Freeze"
0x0: "Timer 16 is not disabled when Core 1 is halted."
0x1: "Timer 16 is disabled when Core 1 is halted."

[24] **SDADC1_DBG_FRZ**: "SDADC 1 Debug Freeze"
0x0: "Timer 16 is not disabled when Core 1 is halted."
0x1: "Timer 16 is disabled when Core 1 is halted."

[23:19] **RESERVED2**: "This field is reserved. It must be kept to zero by the software."

[18] **TIM16_PWM_FRZ**: "Timer 16 PWM Debug Freeze"
0x0: "Timer 16 is not disabled when Core 1 is halted."
0x1: "Timer 16 is disabled when Core 1 is halted."

[17] **TIM15_PWM_FRZ**: "Timer 15 PWM Debug Freeze"
0x0: "Timer 15 is not disabled when Core 1 is halted."
0x1: "Timer 15 is disabled when Core 1 is halted."

[16] **TIM5_PWM_FRZ**: "Timer 5 PWM Debug Freeze"
0x0: "Timer 5 is not disabled when Core 1 is halted."
0x1: "Timer 5 is disabled when Core 1 is halted."

[15] **TIM4_PWM_FRZ**: "Timer 4 PWM Debug Freeze"
0x0: "Timer 4 is not disabled when Core 1 is halted."
0x1: "Timer 4 is disabled when Core 1 is halted."

[14:13] **RESERVED1**: "This field is reserved. It must be kept to zero by the software."

[12] **TIM8_PWM_FRZ**: "Timer 8 PWM Debug Freeze"
0x0: "Timer 8 is not disabled when Core 1 is halted."
0x1: "Timer 8 is disabled when Core 1 is halted."

[11] **TIM1_PWM_FRZ**: "Timer 1 PWM Debug Freeze"
0x0: "Timer 1 is not disabled when Core 1 is halted."
0x1: "Timer 1 is disabled when Core 1 is halted."

[10:4] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[3] **TSU_CAN4_DIS_MORD**: "FDCAN TSU 4 Disable Modify Read"
0x0: "See FDCAN TSU 1 Disable Modify Read."
0x1: "See FDCAN TSU 1 Disable Modify Read."

[2] **TSU_CAN3_DIS_MORD**: "FDCAN TSU 3 Disable Modify Read"
0x0: "See FDCAN TSU 1 Disable Modify Read."
0x1: "See FDCAN TSU 1 Disable Modify Read."

[1] **TSU_CAN2_DIS_MORD**: "FDCAN TSU 2 Disable Modify Read"
0x0: "See FDCAN TSU 1 Disable Modify Read."
0x1: "See FDCAN TSU 1 Disable Modify Read."

[0] **TSU_CAN1_DIS_MORD**: "FDCAN TSU 1 Disable Modify Read"
0x0: "FDCAN TSU 1 modify read is not disabled when Core 1 is halted."
0x1: "FDCAN TSU 1 modify read is disabled when Core 1 is halted: it means that theTSU register bit fields that change after a read operation in normal mode are frozen in debug mode. Therefore a read operation of this register does not change the content."

AXI_C1_FZ_REG AXI Debug Freeze Register Core 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED0																														HRTIMER2_FRZ	HRTIMER1_FRZ
R																														RW	RW

Address: DBGMCUBaseAddress + 0x0008

Type: RW

Reset: 0x0000

Description: This register is used to enable the debug freeze feature of AXI memory mapped IPs for the core 1. This register can be written and read only if DBGEN is active - DBGMCU invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:2] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[1] **HRTIMER2_FRZ**: "HRTimer 2 Debug Freeze"
0x0: "HRTimer 2 is not disabled when Core 1 is halted."
0x1: "HRTimer 2 is disabled when Core 1 is halted."

[0] **HRTIMER1_FRZ**: "HRTimer 1 Debug Freeze"
0x0: "HRTimer 1 is not disabled when Core 1 is halted."
0x1: "HRTimer 1 is disabled when Core 1 is halted."

APB1_C2_FZ_REG APB1 Debug Freeze Register Core 2

31	30 29 28 27 26 25 24 23	22	21	20	19	18	17	16	15	14 13 12 11	10	9 8 7 6	5	4	3 2	1	0
FCCU_DBG	RESERVED4	I2C2_FRZ	I2C1_FRZ	RESERVED3	TIM_TS_FRZ	TIM7_FRZ	TIM6_FRZ	TIM3_FRZ	TIM2_FRZ	RESERVED2	RTC_FRZ	RESERVED1	IWDG2_FRZ	IWDG1_FRZ	RESERVED0	WWDG2_FRZ	WWDG1_FRZ
RW	R	RW	RW	R	RW	RW	RW	RW	RW	R	RW	R	RW	RW	R	RW	RW

Address: DBGMCUBaseAddress + 0x0010

Type: RW

Reset: 0x0000

Description: This register is used to enable the debug freeze feature of APB1 memory mapped IPs for the core 2. This register can be written and read only if DBGEN is active - DBGMCU invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31] **FCCU_DBG**: "FCCU Debug Mode"
0x0: "FCCU Normal Operation"
0x1: "FCCU in debug mode when core 2 is halted. When FCCU_CTRL[DEBUG] is asserted and this bit is set, FCCU moves into debug state and suspends the transition of the FSM by suspending timer elapse time effect."

[30:23] **RESERVED4**: "This field is reserved. It must be kept to zero by the software."

[22] **I2C2_FRZ**: "I2C 2 Debug Freeze"
0x0: "I2C 2 is not disabled when Core 2 is halted."
0x1: "I2C 2 is disabled when Core 2 is halted."

[21] **I2C1_FRZ**: "I2C 1 Debug Freeze"
0x0: "I2C 1 is not disabled when Core 2 is halted."
0x1: "I2C 1 is disabled when Core 2 is halted."

[20] **RESERVED3**: "This field is reserved. It must be kept to zero by the software."

[19] **TIM_TS_FRZ**: "Timer TS Debug Freeze"
0x0: "Timer TS is not disabled when Core 2 is halted."
0x1: "Timer TS is disabled when Core 2 is halted."

[18] **TIM7_FRZ**: "Timer 7 Debug Freeze"
0x0: "Timer 7 is not disabled when Core 2 is halted."
0x1: "Timer 7 is disabled when Core 2 is halted."

[17] **TIM6_FRZ**: "Timer 6 Debug Freeze"
0x0: "Timer 6 is not disabled when Core 2 is halted."
0x1: "Timer 6 is disabled when Core 2 is halted."

[16] **TIM3_FRZ**: "Timer 3 Debug Freeze"
0x0: "Timer 3 is not disabled when Core 2 is halted."
0x1: "Timer 3 is disabled when Core 2 is halted."

[15] **TIM2_FRZ**: "Timer 2 Debug Freeze"
0x0: "Timer 2 is not disabled when Core 2 is halted."
0x1: "Timer 2 is disabled when Core 2 is halted."

[14:11] **RESERVED2**: "This field is reserved. It must be kept to zero by the software."

[10] **RTC_FRZ**: "RTC Debug Freeze"
0x0: "RTC is not disabled when Core 2 is halted."
0x1: "RTC is disabled when Core 2 is halted."

[9:6] **RESERVED1**: "This field is reserved. It must be kept to zero by the software."

[5] **IWDG2_FRZ**: "Independent WDG 2 Debug Freeze"
0x0: "Independent watchdog 2 is not disabled when Core 2 is halted."
0x1: "Independent watchdog 2 is disabled when Core 2 is halted."

[4] **IWDG1_FRZ**: "Independent WDG 1 Debug Freeze"
0x0: "Independent watchdog 1 is not disabled when Core 2 is halted."
0x1: "Independent watchdog 1 is disabled when Core 2 is halted."

[3:2] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[1] **WWDG2_FRZ**: "Window WDG 2 Debug Freeze"
0x0: "Window watchdog 2 is not disabled when Core 2 is halted."
0x1: "Window watchdog 2 is disabled when Core 2 is halted."

[0] **WWDG1_FRZ**: "Window WDG 1 Debug Freeze"
0x0: "Window watchdog 1 is not disabled when Core 2 is halted."
0x1: "Window watchdog 1 is disabled when Core 2 is halted."

APB2_C2_FZ_REG APB2 Debug Freeze Register Core 1

31	30	29	28	27 26	25	24	23 22 21 20 19	18	17	16	15	14 13	12	11	10 9 8 7 6 5 4	3	2	1	0
FDCAN4_DIS_MORD	FDCAN3_DIS_MORD	FDCAN2_DIS_MORD	FDCAN1_DIS_MORD	RESERVED3	SDADC2_DBG_FRZ	SDADC1_DBG_FRZ	RESERVED2	TIM16_PWM_FRZ	TIM15_PWM_FRZ	TIM5_PWM_FRZ	TIM4_PWM_FRZ	RESERVED1	TIM8_PWM_FRZ	TIM1_PWM_FRZ	RESERVED0	TSU_CAN4_DIS_MORD	TSU_CAN3_DIS_MORD	TSU_CAN2_DIS_MORD	TSU_CAN1_DIS_MORD
RW	RW	RW	RW	R	RW	RW	R	RW	RW	RW	RW	R	RW	RW	R	RW	RW	RW	RW

Address: DBGMCUBaseAddress + 0x0014

Type: RW

Reset: 0x0000

Description: This register is used to enable the debug freeze feature of APB2 memory mapped IPs for the core 2. This register can be written and read only if DBGEN is active - DBGMCU invasive debug.
access_size: 32

min_write_access_size: 32
min_read_access_size: 32

[31] **FDCAN4_DIS_MORD**: "FDCAN 4 Disable Modify Read"
0x0: "See FDCAN 1 Disable Modify Read."
0x1: "See FDCAN 1 Disable Modify Read."

[30] **FDCAN3_DIS_MORD**: "FDCAN 3 Disable Modify Read"
0x0: "See FDCAN 1 Disable Modify Read."
0x1: "See FDCAN 1 Disable Modify Read."

[29] **FDCAN2_DIS_MORD**: "FDCAN 2 Disable Modify Read"
0x0: "See FDCAN 1 Disable Modify Read."
0x1: "See FDCAN 1 Disable Modify Read."

[28] **FDCAN1_DIS_MORD**: "FDCAN 1 Disable Modify Read"
0x0: "FDCAN modify read is not disabled when Core 2 is halted."
0x1: "FDCAN modify read is disabled when Core 2 is halted: it means that the register bit fields that change after a read operation in normal mode are frozen in debug mode. Therefore a read operation of this register does not change the content."

[27:26] **RESERVED3**: "This field is reserved. It must be kept to zero by the software."

[25] **SDADC2_DBG_FRZ**: "SDADC 2 Debug Freeze"
0x0: "Timer 16 is not disabled when Core 2 is halted."
0x1: "Timer 16 is disabled when Core 2 is halted."

[24] **SDADC1_DBG_FRZ**: "SDADC 1 Debug Freeze"
0x0: "Timer 16 is not disabled when Core 2 is halted."
0x1: "Timer 16 is disabled when Core 2 is halted."

[23:19] **RESERVED2**: "This field is reserved. It must be kept to zero by the software."

[18] **TIM16_PWM_FRZ**: "Timer 16 PWM Debug Freeze"
0x0: "Timer 16 is not disabled when Core 2 is halted."
0x1: "Timer 16 is disabled when Core 2 is halted."

[17] **TIM15_PWM_FRZ**: "Timer 15 PWM Debug Freeze"
0x0: "Timer 15 is not disabled when Core 2 is halted."
0x1: "Timer 15 is disabled when Core 2 is halted."

[16] **TIM5_PWM_FRZ**: "Timer 5 PWM Debug Freeze"
0x0: "Timer 5 is not disabled when Core 2 is halted."
0x1: "Timer 5 is disabled when Core 2 is halted."

[15] **TIM4_PWM_FRZ**: "Timer 4 PWM Debug Freeze"
0x0: "Timer 4 is not disabled when Core 2 is halted."
0x1: "Timer 4 is disabled when Core 2 is halted."

[14:13] **RESERVED1**: "This field is reserved. It must be kept to zero by the software."

[12] **TIM8_PWM_FRZ**: "Timer 8 PWM Debug Freeze"
0x0: "Timer 8 is not disabled when Core 2 is halted."
0x1: "Timer 8 is disabled when Core 2 is halted."

[11] **TIM1_PWM_FRZ**: "Timer 1 PWM Debug Freeze"
0x0: "Timer 1 is not disabled when Core 2 is halted."
0x1: "Timer 1 is disabled when Core 2 is halted."

[10:4] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[3] **TSU_CAN4_DIS_MORD**: "FDCAN TSU 4 Disable Modify Read"
0x0: "See FDCAN TSU 1 Disable Modify Read."
0x1: "See FDCAN TSU 1 Disable Modify Read."

[2] **TSU_CAN3_DIS_MORD**: "FDCAN TSU 3 Disable Modify Read"
0x0: "See FDCAN TSU 1 Disable Modify Read."
0x1: "See FDCAN TSU 1 Disable Modify Read."

[1] **TSU_CAN2_DIS_MORD**: "FDCAN TSU 2 Disable Modify Read"
0x0: "See FDCAN TSU 1 Disable Modify Read."
0x1: "See FDCAN TSU 1 Disable Modify Read."

[0] **TSU_CAN1_DIS_MORD**: "FDCAN TSU 1 Disable Modify Read"
0x0: "FDCAN TSU 1 modify read is not disabled when Core 2 is halted."
0x1: "FDCAN TSU 1 modify read is disabled when Core 2 is halted: it means that theTSU register bit fields that change after a read operation in normal mode are frozen in debug mode. Therefore a read operation of this register does not change the content."

AXI_C2_FZ_REG AXI Debug Freeze Register Core 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED0																														HRTIMER2_FRZ	HRTIMER1_FRZ
R																														RW	RW

Address: DBGMCUBaseAddress + 0x0018

Type: RW

Reset: 0x0000

Description: This register is used to enable the debug freeze feature of AXI memory mapped IPs for the core 2. This register can be written and read only if DBGEN is active - DBGMCU invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:2] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[1] **HRTIMER2_FRZ**: "HRTimer 2 Debug Freeze"
0x0: "HRTimer 2 is not disabled when Core 2 is halted."
0x1: "HRTimer 2 is disabled when Core 2 is halted."

[0] **HRTIMER1_FRZ**: "HRTimer 1 Debug Freeze"
0x0: "HRTimer 1 is not disabled when Core 2 is halted."
0x1: "HRTimer 1 is disabled when Core 2 is halted."

DBG_RST_REG Debug Force Reset Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														FPD_DEST	FPD_FUNC
R																														RW	RW

Address: DBGMCUBaseAddress + 0x0020

Type: RW

Reset: 0x0000

Description: This register is used to perform both functional and destructive reset. It is replacing DCI jtag reset source defined for other M40 devices. This register can be written and read only if DBGEN is active - DBGMCU invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[1] **FPD_DEST**: "Force Destructive Reset Bit. This field must be written and cleared by the software."
0x0: "Do not force destructive reset"
0x1: "Force destructive reset."

[0] **FPD_FUNC**: "Force Functional Reset Bit. This field must be written and cleared by the software."
0x0: "Do not force functional reset"
0x1: "Force functional reset."

DOUT_REG DBGMCU Output Data Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT																															
RW																															

Address: DBGMCUBaseAddress + 0x0024

Type: RW

Reset: 0x0000

Description: This register can be written by the software to implement the JTAG Password or Challenge/Response authentication protocol. It is replacing JOUT_IPS JDC register. This register can be only written by the cores while it is read-only for the external tool.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **DOUT**: "This register can be written by the software to implement the JTAG Password or Challenge/Response authentication protocol. This register can be only written by the secure master and it is RO for the other masters."

DIN_REG DBGMCU Input Data Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN																															
RW																															

Address: DBGMCUBaseAddress + 0x0028

Type: RW

Reset: 0x0000

Description: This register can be written by the external tool to implement the JTAG Password or Challenge/Response authentication protocol. It is replacing JIN_IPS JDC register. This register can be only written by the external tool while it is read-only for the other masters.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **DIN**: "This register can be written by the external tool to implement the JTAG Password or Challenge/Response authentication protocol. This register can be only written by the external tool and it is RO for the other masters."

DMCR DBGMCU Module Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED1															DIN_IE	RESERVED0															DOUT_IE
R															RW	R															RW

Address: DBGMCUBaseAddress + 0x002C

Type: RW

Reset: 0x0000

Description: This register is can be written to enable DIN and DOUT interrupt. It is replacing MCR JDC register. This register is always accessible.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:17] **RESERVED1**: "This field is reserved. It must be kept to zero by the software."

[16] **DIN_IE**: "This bit is written and cleared by the software. It is used to enable or disable DIN interrupt flag."

0x0: "DIN interrupt flag is disabled."

0x1: "DIN interrupt flag is enabled. An interrupt is generated if a new data is received in DIN register."

[15:1] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[0] **DOUT_IE**: "This bit is written and cleared by the software. It is used to enable or disable DOUT interrupt flag."

0x0: "DOUT interrupt flag is disabled."

0x1: "DOUT interrupt flag is enabled. An interrupt is generated if a new data is received in DOUT register."

DMSR DBGMCU Module Status Register

31	30 29 28 27 26 25	24	23 22 21 20 19 18 17	16	15 14 13 12 11 10 9	8	7 6 5 4 3 2 1	0
DEVICE_EN	RESERVED3	DIN_IF	RESERVED2	DIN_RDY	RESERVED1	DOUT_IF	RESERVED0	DOUT_RDY
R	R	RW	R	R	R	RW	R	R

Address: DBGMCUBaseAddress + 0x0030

Type: RW

Reset: 0x0000

Description: This register shows when DOUT and DIN data are ready in the corresponding registers. It also shows if the interrupt flags for DOUT and DIN data are generated. It is replacing MSR JDC register. This register is always accessible.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31] **DEVICE_EN**: "This bit is set if DBGEN&NIDEN = 1; it means DBGMCU is in full debug."

[30:25] **RESERVED3**: "This field is reserved. It must be kept to zero by the software."

[24] **DIN_IF**: "This bit is set if a new data is received in the DIN_REG and the correspondent bit in the DMCR is set. It must be cleared by the software writing this bit field to 0x1."

[23:17] **RESERVED2**: "This field is reserved. It must be kept to zero by the software."

[16] **DIN_RDY**: "This bit is set if a new data is ready in the DIN_REG. It is automatically cleared by the hardware when DIN_REG is read."

[15:9] **RESERVED1**: "This field is reserved. It must be kept to zero by the software."

[8] **DOUT_IF**: "This bit is set if a new data is received in the DOUT_REG and the correspondent bit in the DMCR is set. It must be cleared by the software writing this bit field to 0x1."

[7:1] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[0] **DOUT_RDY**: "This bit is set if a new data is ready in the DOUT_REG. It is automatically cleared by the hardware when DOUT_REG is read."

CTI_TRG_SEL CTI Trigger Selection Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED1											TODBGENSEL					RESERVED0												TINIDENSEL			
R											RW					R												RW			

Address: DBGMCUBaseAddress + 0x0040

Type: RW

Reset: 0x0000

Description: This register is written and cleared by the software; it can be used to configure TINDENSEL and TODGBENSEL of system CTI. This register can be written and read only if DBGEN is active - DBGMCU invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:21] **RESERVED1**: "This field is reserved. It must be kept to zero by the software."

[20:16] **TODBGENSEL**: "When dbgen (CTI) is HIGH, all event_out[n] outputs are enabled. When dbgen is LOW:
- event_out[n] is enabled if todbgensel[n] is HIGH
- event_out[n] is masked if todbgensel[n] is LOW"

[15:4] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[3:0] **TINIDENSEL**: "When niden (CTI) is HIGH, all event_in[n] inputs are enabled. When niden is LOW:
- event_in[n] is enabled if tinidensel[n] is HIGH
- event_in[n] is masked if tinidensel[n] is LOW"

DBG_SLEEP_CFG DBGMCU Debug Sleep Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED1															DBG_SLEEPH	RESERVED0														DBG_SLEEP2	DBG_SLEEP1
R															RW	R														RW	RW

Address: DBGMCUBaseAddress + 0x0044

Type: RW

Reset: 0x0000

Description: This register can be written to set the DBG_SLEEP bit for Core 1, Core 2 and Core H
access_size: 32

min_write_access_size: 32
min_read_access_size: 32

[31:17] **RESERVED1**: "This field is reserved. It must be kept to zero by the software."

[16] **DBG_SLEEPH**: "This bit is written and cleared by the software. It is used to set the debug sleep for Core H."

0x0: "Normal operation - processor clock is stopped automatically when WFI/WFE are executed."

0x1: "Automatic clock stop disabled - processor clock continues to run, allowing full debug capability."

[15:2] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[1] **DBG_SLEEP2**: "This bit is written and cleared by the software. It is used to set the debug sleep for Core 2."

0x0: "Normal operation - processor clock is stopped automatically when WFI/WFE are executed."

0x1: "Automatic clock stop disabled - processor clock continues to run, allowing full debug capability."

[0] **DBG_SLEEP1**: "This bit is written and cleared by the software. It is used to set the debug sleep for Core 1."

0x0: "Normal operation - processor clock is stopped automatically when WFI/WFE are executed."

0x1: "Automatic clock stop disabled - processor clock continues to run, allowing full debug capability."

DEVARCH Device Architecture Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEV_ARCH																															
R																															

Address: DBGMCUBaseAddress + 0x0FBC

Type: R

Reset: 0x0410 0000

Description: CS Device Architecture Register. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **DEV_ARCH**: "This field is used to identifies the architect and the architecture of the CS component."

DEVTYPE Device Type Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED0																								DEV_TYPE							
R																								R							

Address: DBGMCUBaseAddress + 0x0FCC

Type: R

Reset: 0x0004

Description: CS Device Type Register. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:8] **RESERVED0**: "This field is reserved. It must be kept to zero by the software."

[7:0] **DEV_TYPE**: "This field is used to identifies the type of CS component: 0x4 to identifies a generic debug control component."

PIDR4 Peripheral Identification Register 4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR4_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FD0

Type: R

Reset: 0x0000

Description: CS Peripheral Identification Register 4. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **PIDR4_VAL**: "Peripheral Identification Register 4"

PIDR5 Peripheral Identification Register 5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR5_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FD4

Type: R

Reset: 0x0000

Description: CS Peripheral Identification Register 5. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **PIDR5_VAL**: "Peripheral Identification Register 5"

PIDR6 Peripheral Identification Register 6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR6_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FD8

Type: R

Reset: 0x0000

Description: CS Peripheral Identification Register 6. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **PIDR6_VAL**: "Peripheral Identification Register 6"

PIDR7 Peripheral Identification Register 7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR7_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FDC

Type: R

Reset: 0x0000

Description: CS Peripheral Identification Register 7. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **PIDR7_VAL**: "Peripheral Identification Register 7"

PIDR0 Peripheral Identification Register 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR0_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FE0

Type: R

Reset: 0x0000

Description: CS Peripheral Identification Register 0. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **PIDR0_VAL**: "Peripheral Identification Register 0"

PIDR1 Peripheral Identification Register 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR1_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FE4

Type: R

Reset: 0x0000

Description: CS Peripheral Identification Register 1. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **PIDR1_VAL**: "Peripheral Identification Register 1"

PIDR2 Peripheral Identification Register 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR2_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FE8

Type: R

Reset: 0x000A

Description: CS Peripheral Identification Register 2. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32

min_write_access_size: 32
min_read_access_size: 32

[31:0] **PIDR2_VAL**: "Peripheral Identification Register 2"

PIDR3 Peripheral Identification Register 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR3_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FEC

Type: R

Reset: 0x0000

Description: CS Peripheral Identification Register 3. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **PIDR3_VAL**: "Peripheral Identification Register 3"

CIDR0 Component Identification Register 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR0_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FF0

Type: R

Reset: 0x000D

Description: CS Component Identification Register 0. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **CIDR0_VAL**: "Component Identification Register 0"

CIDR1 Component Identification Register 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR1_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FF4

Type: R

Reset: 0x0090

Description: CS Component Identification Register 1. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **CIDR1_VAL**: "Component Identification Register 1"

CIDR2 Component Identification Register 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR2_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FF8

Type: R

Reset: 0x0005

Description: CS Component Identification Register 2. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **CIDR2_VAL**: "Component Identification Register 2"

CIDR3 Component Identification Register 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR3_VAL																															
R																															

Address: DBGMCUBaseAddress + 0x0FFC

Type: R

Reset: 0x00B1

Description: CS Component Identification Register 3. See CoreSight Architecture Spec for further details. This register can be read only if NIDEN is active - DBGMCU non-invasive debug.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:0] **CIDR3_VAL**: "Component Identification Register 3"

11 Reset and clock control (RCC)

11.1 Reset

There are two types of reset, defined as destructive reset and functional reset.

Refer to *Chapter 6: Reset and boot* for detailed information.

The reset source can be identified by checking the flags in the destructive event status register (DESR) and functional event status register (FESR).

11.2 Clocks

Five different clock sources can be used to drive the system clock (SYSCLK):

- Internal RC oscillator 16 MHz (IRCOSC)
- External crystal oscillator 4-40 MHz (XOSC)
- PLL0 clock
- PLL1 clock
- SSCG - Speed Spectrum Clock Generator (FM)

The IRCOSC is used as system clock source after startup from Reset.

The devices have the following additional clock sources: 1 MHz low speed internal RCOSC (LSIRCOSC) which can drive the RTC and the UARTs.

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Fixed prescalers are configuring the frequencies of the AHB, APB1 and APB2 domains.

All the peripheral kernel clocks are derived from their bus clock (HCLK, PCLK1 or PCLK2) except:

- The ADCs clock which is derived (selected by software) from one of the following sources:
  - PLL0 clock
  - PLL1 clock
  - XOSC clock
- The UARTs clocks which are derived (selected by software) from one of the four following sources:
  - PLL0
  - IRCOSC clock
  - XOSC clock
  - LSICLK clock
- The I^2Cs clocks which are derived (selected by software) from one of the three following sources:
  - PLL0 clock
  - IRCOSC clock
  - XOSC clock

- The RTC clock which is derived (selected by software) from one of the following sources:
  - LSICLK clock
  - XOSC clock divided by 32
- The IWDG clock which is always the IRCOSC clock.
- The FDCAN1 clock, which is derived (selected by software) from one of the two following sources:
  - XOSC clock
  - PLL0 clock

The FCCU clock is always the IRCOSC clock. The STCU clock is always SYSCLK clock.

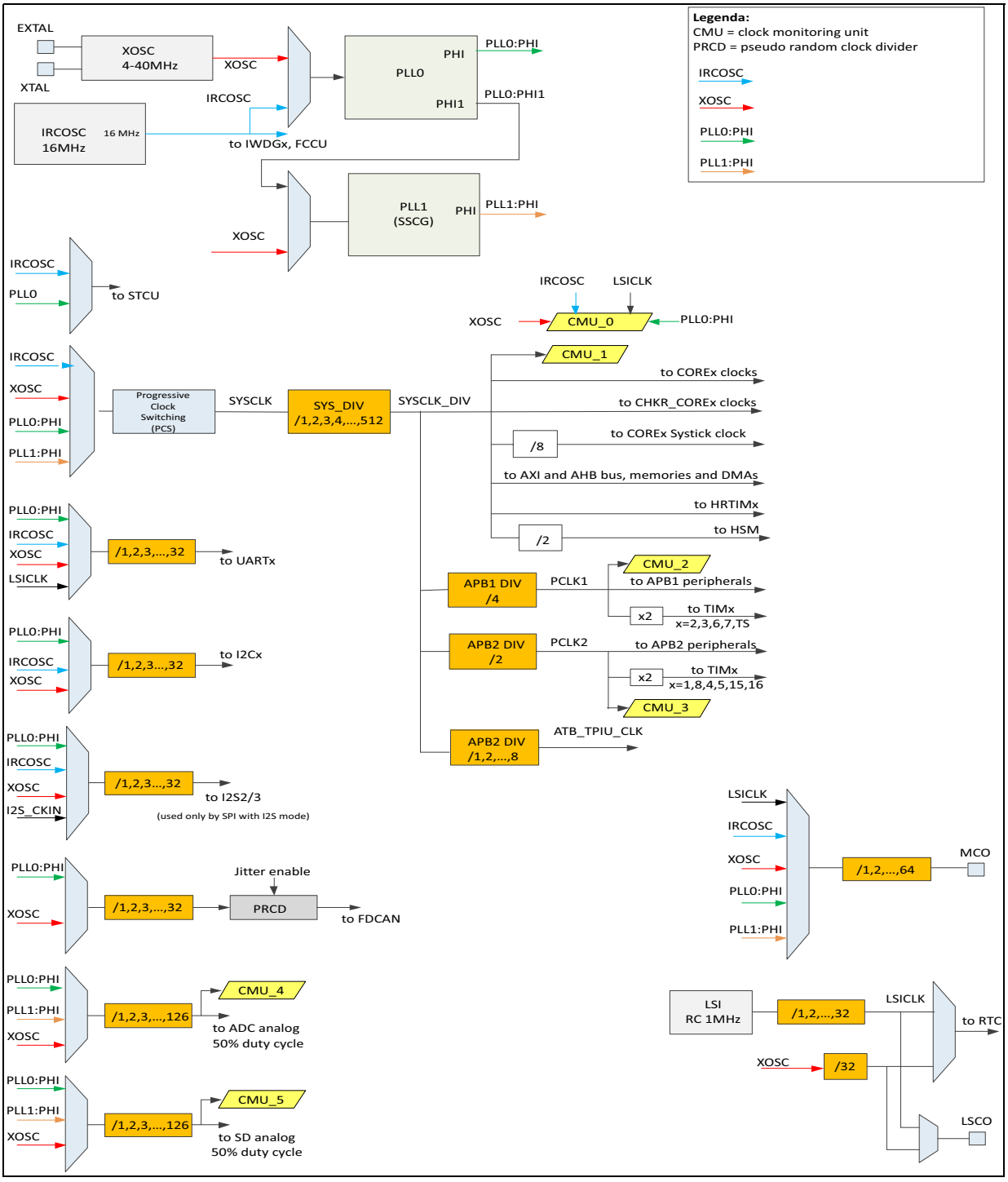
The RCC feeds the Cortex® System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or directly with the Cortex® clock (HCLK), configurable in the SysTick Control and Status Register.

FCLK acts as Cortex®-M7 free-running clock. For more details refer to the Cortex®-M7 technical reference manual (see http://infocenter.arm.com).

Table 108. Maximum IP clock frequencies

IP clock	Peripherals	Clock selector	Max division factor	Division factor at reset	Max frequency (MHz)
UART_CLK	UART	UART_SEL	32	3	100
I2C_CLK	I2C	I2C_SEL	32	3	100
I2S_CLK	I2S	I2S_SEL	32	3	100
FDCAN_CLK	FDCAN	FDCAN_SEL	32	4	80
ADC_CLK	ADC	ADC_SEL	126	6	40
SD_ADC_CLK	SD_ADC	SD_ADC_SEL	126	18	16
HRTIM_CLK	HRTIM	System clock switch	—	—	FSYS (max)
APB1_CLK	APB1	—	—	—	FSYS (max) / 4
APB2_CLK	APB2	—	—	—	FSYS (max) / 2
TIMx_CLK x = 2, 3, 6, 7, TS	TIMx x = 2, 3, 6, 7, TS	—	—	—	FSYS (max) / 2
TIMy_CLK y = 1, 4, 5, 8, 15, 16	TIMy y = 1, 4, 5, 8, 15, 16	—	—	—	FSYS (max)
HSM_CLK	HSM	—	—	—	FSYS (max) / 2

Figure 47. Clock tree



Note: For full details about the internal and external clock source characteristics, refer to the "Electrical characteristics" section in the device datasheet.

The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is '1', the AHB prescaler must be equal to '1'.

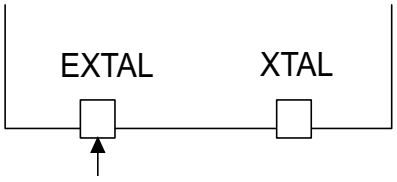
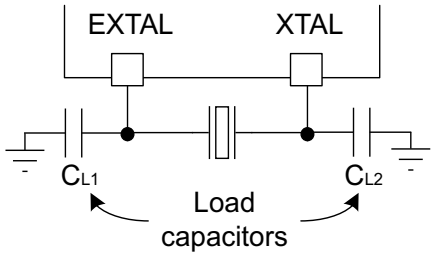
11.2.1 External oscillator (XOSC)

The external crystal oscillator clock signal (XOSC) can be generated from two possible clock sources:

- XOSC external crystal
- XOSC user external clock

The external crystal and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

Table 109. XOSC clock sources

Clock source	Hardware configuration
External clock	 MSv31915V1
Crystal	 MSv31916V1

External crystal

The 4 to 40 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in *Table 109*. Refer to the electrical characteristics section of the *datasheet* for more details.

The XOSCRDY flag in the *Clock Sources Control Register* (*CR*) indicates if the XOSC oscillator is stable or not. Once enabled, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock Interrupt / Fault Enable* (*CIER*).

The XOSC Crystal can be switched on and off using the XOSCON bit in the *Clock Sources Control Register* (*CR*).

External source (bypass)

In this mode, an external clock source must be provided. You select this mode by setting the XOSCBYP and XOSCON bits in the *Clock Sources Control Register* (*CR*). The external clock signal has to drive the EXTAL pin (refer to the datasheet for the proper values). See *Table 109*.

Automatic level control (ALC)

The function of the ALC block is to provide high drive current during oscillator start up, and reduce current after oscillation to reduce power, distortion, RFI, and to avoid overdriving the crystal. ALC block can be controlled with DCF by configuring the UTEST miscellaneous bit 14 [XOSC_ALC_DIS]. Comparing the configurations with ALC disabled against the case with ALC enabled, there are the following attributes:

- Higher oscillation amplitudes at crystal.
- Distorted oscillation with several harmonics.
- Increased EMI.
- Higher crystal drive level.
- Less susceptible to noise.

The ALC enable configuration is suitable for most applications. The amplitude of oscillations is lower, hence there are less distortion in oscillations, less EMI, and less crystal drive level.

Feedback resistor is integrated. It does not require external resistors by offering a lower cost solution and simpler PCB design.

Note: It is recommended to make ALC feature enabled (that is, XOSC_ALC_DIS = 0) for low dissipation in crystal (crystal drive level).

11.2.2 Internal RC oscillator (IRCOSC)

The IRCOSC clock signal is generated from an internal 16 MHz RC Oscillator.

The IRCOSC RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the XOSC crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator.

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated. Refer to the device datasheet for calibration accuracy.

After reset, the factory calibration value is loaded in the IRCOSC_TRIM[7:0] bits in the *Internal Clock Sources Calibration Register* (*ICSR*).

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. You can trim the IRCOSC frequency in the application using the IRCOSC_TRIM_USER[6:0] in the *Internal Clock Sources Calibration Register* (*ICSR*).

For more details on how to measure the IRCOSC frequency variation, refer to *Section 11.2.13: Internal/external clock measurement with TIM5/TIM15/TIM16*.

The IRCOSCRDY flag in the *Clock Sources Control Register* (*CR*) indicates if the IRCOSC oscillator is stable or not. At startup, the IRCOSC output clock is not released until this bit is set by hardware.

Frequency trimming

In order to compensate for the process variation of Resistor and Capacitor, an 8-bit frequency trimming is used. The pins used for this purpose are TRIM_BIT<7:0>. Out of these, TRIM_BIT<7:5> are used as course trimming bits, which are provided as input to cap- trimming circuitry. Further, TRIM_BIT<4:0> are used as fine trimming bits, which are provided as input to res-trimming circuitry. Along with TRIM_BIT<7:0> user is also provided with a trimming capability. TRIM_BIT_USER<4:0> is in user control to trim the frequency.

Table 110. Course trimming

S. No.	TRIM_BIT<7:5>	Frequency
1	100	F + 62.0%
2	101	F + 44.1%
3	110	F + 28.2%
4	111	F + 12.8%
5	000	F
6	001	F - 11.2%
7	010	F - 21.8%
8	011	F - 30.9%

Table 111. FINE trimming

S. No.	TRIM_BIT<4:0>	Frequency
1	10000	F + 11.1%
2	10001	F + 10.3%
3	10010	F + 9.6%
4	10011	F + 8.8%
5	10100	F + 8.1%

Table 111. FINE trimming (continued)

S. No.	TRIM_BIT<4:0>	Frequency
6	10101	F + 7.3%
7	10110	F + 6.6%
8	10111	F + 5.9%
9	11000	F + 5.2%
10	11001	F + 4.5%
11	11010	F + 3.8%
12	11011	F + 3.2%
13	11100	F + 2.5%
14	11101	F + 1.8%
15	11110	F + 1.2%
16	11111	F + 0.6%
17	00000	F
18	00001	F - 0.6%
19	00010	F - 1.2%
20	00011	F - 1.8%
21	00100	F - 2.4%
22	00101	F - 3.0%
23	00110	F - 3.6%
24	00111	F - 4.2%
25	01000	F - 4.7%
26	01001	F - 5.3%
27	01010	F - 5.9%
28	01011	F - 6.4%
29	01100	F - 7.0%
30	01101	F - 7.5%
31	01110	F - 8.1%
32	01111	F - 8.6%
31	01110	F - 8.1%
32	01111	F - 8.6%

When simulated at Typical Corner of DK_cmosm40_1.1.2, at AVDD12=1.2 V and Temperature of 27 °C, the frequency at TRIM_BIT<7:0> = <00000000> was found to be 14.828 MHz. This value is subject to process, supply and temperature values. In order to guarantee frequency trimmability across process, default value around 15 MHz has been chosen with respect to nominal 16 MHz.

User trimming

User trim code consists of two parts, MSB represents the sign bit (current source or sink) while TRIM_BIT_USER<4:0> represents the actual code. If MSB is kept zero, then increasing the TRIM_BIT_USER<4:0> by one step decreases the frequency stepwise. Similarly if MSB is kept 1, then increasing the TRIM_BIT_USER<4:0> by one step increases the frequency stepwise.

Table 112 summarizes the expected variation on output frequency.

Table 112. USER trimming

S. No.	TRIM_BIT_USER<5:0>	Frequency
-32	100000	F1+5.16%
-31	100001	F1+5.16%
-30	100010	F1+5.16%
-29	100011	F1+5.16%
-28	100100	F1+5.16%
-27	100101	F1+5.16%
-26	100110	F1+5.16%
-25	100111	F1+5.16%
-24	101000	F1+5.16%
-23	101001	F1+5.16%
-22	101010	F1+5.16%
-21	101011	F1+5.16%
-20	101100	F1+5.16%
-19	101101	F1+5.16%
-18	101110	F1+5.16%
-17	101111	F1+5.16%
-16	110000	F1+5.16%
-15	110001	F1+4.83%
-14	110010	F1+4.50%
-13	110011	F1+4.17%
-12	110100	F1+3.84%
-11	110101	F1+3.51%
-10	110110	F1+3.19%
-9	110111	F1+2.86%
-8	111000	F1+2.54%
-7	111001	F1+2.22%
-6	111010	F1+1.89%
-5	111011	F1+1.57%

Table 112. USER trimming (continued)

S. No.	TRIM_BIT_USER<5:0>	Frequency
-4	111100	F1+1.25%
-3	111101	F1+0.94%
-2	111110	F1+0.62%
-1	111111	F1+0.31%
0	000000	F1=16MHz
1	000001	F1-0.29%
2	000010	F1-0.58%
3	000011	F1-0.87%
4	000100	F1-1.17%
5	000101	F1-1.45%
6	000110	F1-1.74%
7	000111	F1-2.02%
8	001000	F1-2.30%
9	001001	F1-2.59%
10	001010	F1-2.87%
11	001011	F1-3.15%
12	001100	F1-3.43%
13	001101	F1-3.71%
14	001110	F1-3.98%
15	001111	F1-4.26%
16	010000	F1-4.53%
17	010001	F1-4.53%
18	010010	F1-4.53%
19	010011	F1-4.53%
20	010100	F1-4.53%
21	010101	F1-4.53%
22	010110	F1-4.53%
23	010111	F1-4.53%
24	011000	F1-4.53%
25	011001	F1-4.53%
26	011010	F1-4.53%
27	011011	F1-4.53%
28	011100	F1-4.53%
29	011101	F1-4.53%

Table 112. USER trimming (continued)

S. No.	TRIM_BIT_USER<5:0>	Frequency
30	011110	F1-4.53%
31	011111	F1-4.53%

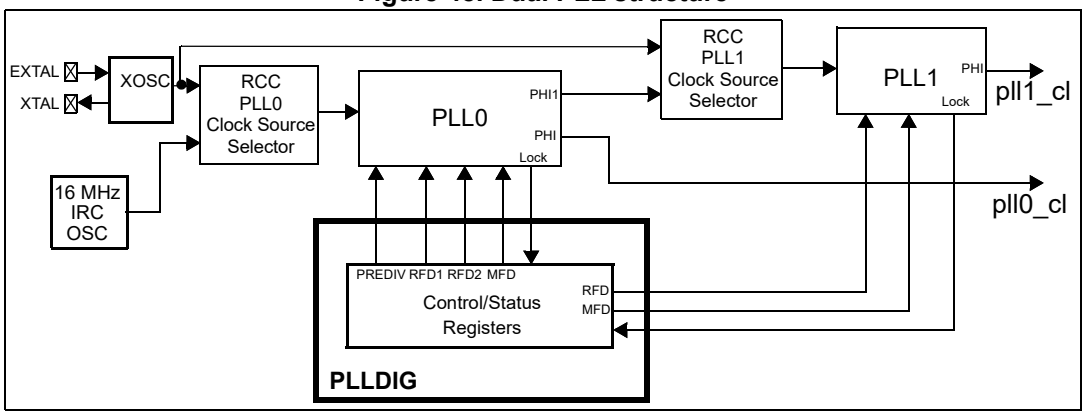
11.2.3 PLL

The Dual PLL provides separate system and peripheral clocks as shown in *Figure 47: Clock tree*.

The PLLs are disabled after power is cycled and must be enabled by software.

The block diagram for the Dual PLL structure is shown in *Figure 48*.

Figure 48. Dual PLL structure



PLL0: base PLL (non-FM)

This primary PLL provides a clock that is not Frequency Modulated to the peripheral IPs and the reference clock to PLL1.

The input clock sources for PLL0 are:

– XOSC
– IRCOSC
– The external reference clock (input on the EXTAL pin)

The output clocks from PLL0 are:

– PHI: drives the peripheral clock and the system clock when PLL1 is not locked or active.
– PHI1: provides the input reference clock for PLL1.

PLL1: FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) that is used to drive the system clock.

The input clocks to PLL1 are:

– PHI1 output clock from PLL0
– XOSC

The output clock from PLL1 is:

– PHI: drives the system clock. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1 PHI output clock frequency.

PLL register interface

This device has a single digital interface for the dual PLLs. The digital interface provides register control of all features of the PLLs. Details are provided in the *Chapter 12: Dual PLL digital interface (PLLDIG)*.

Loss of clock detection

Loss of clock detection must be enabled at all times when the PLLs are enabled.

XOSC loss of Clock event from CMU0 and loss of lock events from PLL0 and PLL1 are routed as fault to FCCU, where there is a software option to generate a reset or interrupt related to each fault condition.

Software may enable or disable an optional backup switch to safe clock in case of XOSC loss of clock or PLLs loss of lock, setting the corresponding bits in *CRRCR*.

PLLs initialization sequence

Refer to *Section 12.7: Initialization information* in *Chapter 12: Dual PLL digital interface (PLLDIG)* for the initialization sequence of PLL0 and PLL1 at device start-up.

11.2.4 Low speed internal RC oscillator (LSIRCOSC)

The LSI RC can clock the RTC through a programmable prescaler. The clock frequency is 1 MHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI RC can be switched on and off using the LSION bit in the *Clock Sources Control Register* (*CR*).

The LSIRDY flag in the *Clock Sources Control Register* (*CR*) indicates if the LSI oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock Interrupt / Fault Enable* (*CIER*).

11.2.5 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

– IRCOSC oscillator
– XOSC oscillator
– PLL0
– PLL1

Note: Refer to SR5E1x Datasheet for actual electrical parameters and characteristics.

After a system reset, the IRCOSC oscillator is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *Internal Clock Sources Calibration Register* (*ICSR*) indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

To switch from low speed to high speed or from high speed to low speed system clock, it is recommended to use a transition state with medium speed clock, for at least 1 µs.

Clock source switching conditions:

- Switching from XOSC or IRCOSC to PLL with AHB frequency (HCLK) higher than 80 MHz
- Switching from PLL with HCLK higher than 80 MHz to XOSC or IRCOSC

Transition state:

- Set the AHB prescaler HPRE[3:0] bits to divide the system frequency by 2
- Switch system clock to PLL
- Wait for at least 1 µs and then reconfigure AHB prescaler bits to the needed HCLK frequency

11.2.6 System clock switching to IRCOSC

In case of unrecoverable failures involving the clocks that can bring to the loss of system clock, the main system clock can be switched on IRCOSC.

The RCC_CRRCR allows to control which event can generate a Switching to IRCOSC clock. Default value is disabled for each event.

Switching on IRCOSC is routed as fault event to FCCU.

The switch of the system clock to IRCOSC must happen in case one of the following events occurs and if the system clock is derived from the clock source, interested by the fail:

- XOSC loss, signaled by CMU_0. In addition, this signal is routed separately to FCCU too.
- PLL0 loss of lock. In addition, this signal is routed separately to FCCU too.
- PLL1 loss of lock. In addition, this signal is routed separately to FCCU too.
- CMU1 out of range. In addition, this signal is routed separately to FCCU too.

11.2.7 ADC clock

The ADC clock is derived from the XOSC, the PLL0 or the PLL1 via a prescaler. It can reach 306 MHz and can be divided by the following prescalers values: 1, 2, 4, 6, 8, 10,... 126 by configuring the ADC_PRE field of *CCIPR2* register. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bit fields in the ADCx_CCR.

If the programmed factor is ‘1’, the AHB prescaler must be set to ‘1’.

11.2.8 RTC clock

The RTCCLK clock source can be either the XOSC/32 or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the *Low Speed Clock Configuration register* (*LSCFGR*). This selection cannot be modified without resetting the RTC domain. The system must always be configured so as to get a PCLK frequency greater than or equal to the RTCCLK frequency for a proper operation of the RTC.

Note: If LSI is selected as the RTC clock:

The RTC state is not guaranteed if the V_{DD} supply is powered off.

Note: If the XOSC clock divided by a prescaler is used as the RTC clock:

The RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the V_{CORE} domain).

When the RTC clock is LSI, the RTC remains clocked and functional under system reset.

11.2.9 LSICLK clock branch

LSICLK clock is the LSIRCOSC clock divided by a prescaler configured by LSI_PRE field of LSCFGR RCC register.

11.2.10 Timer clock

The timer clock frequencies are automatically defined by hardware. They are set to twice (×2) the frequency of the APB domain.

11.2.11 Watchdog clock

IWDG kernel clock is the IRCOSC that is always on.

11.2.12 Clock-out capability

- M2CO

  The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. One of the five clock signals can be selected as the MCO clock:

  – LSIRCOSC
  – IRCOSC
  – XOSC
  – PLL0
  – PLL1

  The selection is controlled by the MCOSEL[3:0] bits of the *Clock Configuration Register* (*CFGR*). The selected clock can be divided with the MCOPRE[2:0] field of the *Clock Configuration Register* (*CFGR*).

- LSCO

  Another output (LSCO) allows one of the following low speed clocks to be output onto the external LSCO pin:

  – LSI can be divided by its dedicated prescaler
  – XOSC/32

  The selection is controlled by the LSCOSEL, and enabled with the LSCOEN in the *Low Speed Clock Configuration register* (*LSCFGR*).

The MCO clock output requires the corresponding alternate function selected on the MCO pin, the LSCO pin must be left in default POR state.

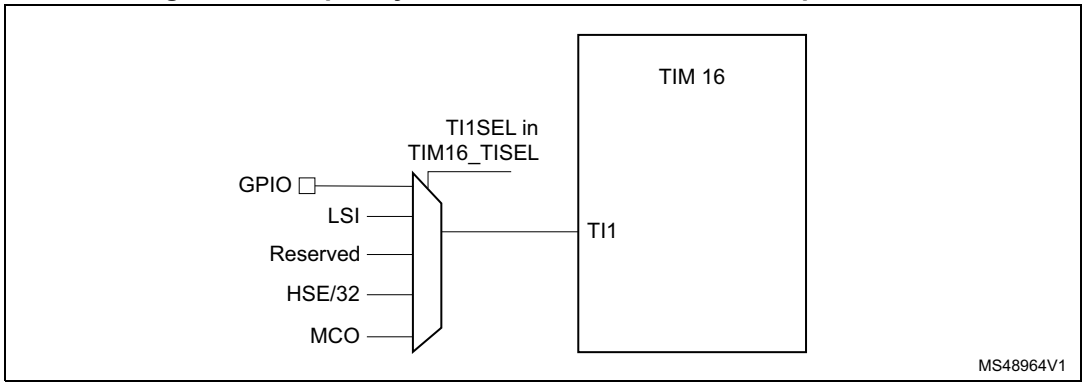
11.2.13 Internal/external clock measurement with TIM5/TIM15/TIM16

It is possible to indirectly measure the frequency of all on-board clock sources by means of the TIM5, TIM15, TIM16 channel 1 input capture, as represented in *Figure 49* and *Figure 50*.

The input capture channel of the Timer 15 can be a GPIO line.

TIM15 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the SR5E1x pinout Microsoft Excel® file attached to the IO_Definition document.

Figure 49. Frequency measurement with TIM16 in capture mode

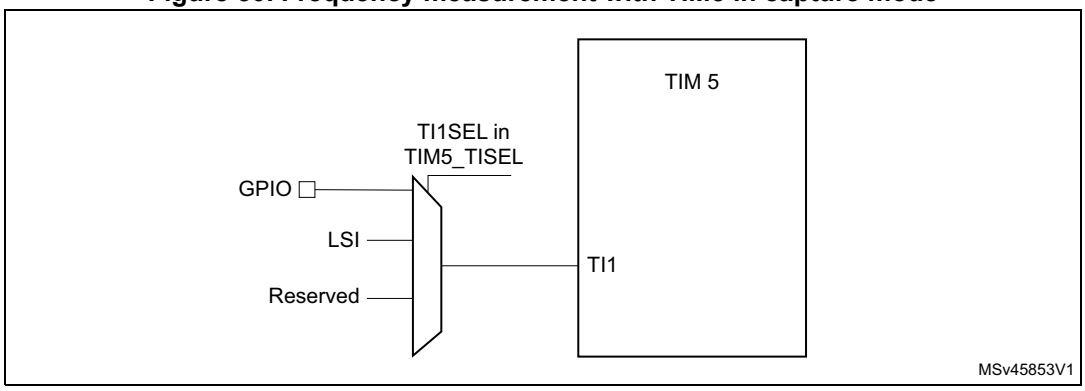


The input capture channel of the Timer 16 can be a GPIO line or an internal clock of the MCU.

The possibilities are the following ones:

- TIM16 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the SR5E1x pinout Microsoft Excel® file attached to the IO_Definition document.
- TIM16 Channel1 is connected to the LSI clock.
- TIM16 Channel1 is connected to the XOSC/32 clock.
- TIM16 Channel1 is connected to the MCO.

Figure 50. Frequency measurement with TIM5 in capture mode



The input capture channel of the Timer 5 can be a GPIO line or an internal clock of the MCU.

The possibilities are the following ones:

- TIM5 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the SR5E1x pinout Microsoft Excel® file attached to the IO_Definition document.
- TIM5 Channel1 is connected to the LSI Clock.

Calibration of the IRCOSC

For TIM16, the primary purpose of connecting the XOSC/32 to the channel 1 input capture is to be able to precisely measure the IRCOSC system clocks (for this, the IRCOSC must be

used as the system clock source). The number of IRCOSC clock counts between consecutive edges of the XOSC/32 signal provides a measure of the internal clock period. Taking advantage of the high precision of XOSC/32 crystals (typically a few tens of ppm's), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing, process, temperature and/or voltage related frequency deviations.

The IRCOSC oscillator has dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (for example the ratio between IRCOSC and XOSC/32): the precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement is.

11.2.14 Peripheral clock enable register (RCC_AHBxENR, RCC_APBxENR)

Each peripheral clock can be enabled by the xxxxEN bit of the RCC_AHBxENR, RCC_APBxENR registers.

When the peripheral clock is not active, the peripheral registers read or write accesses are not supported.

The enable bit has a synchronization mechanism to create a glitch free clock for the peripheral. After the enable bit is set, there is a 2 clock cycles delay before the clock is active.

Caution: Just after enabling the clock for a peripheral, software must wait for a delay before accessing the peripheral registers.

11.3 Clock monitoring

For all safety relevant clocks the microcontroller uses clock monitoring units (CMUs) to detect a missing clock or incorrect frequency.

Each CMU is programmed independently (refer to *Figure 47* for CMU distribution) and uses the IRCOSC or XOSC as the clock monitor reference.

Detailed information on the CMUs can be found in the Clock Monitor Unit chapter.

11.3.1 CMU configuration

This section explains the CMU configuration.

Figure 51 shows the block diagram for CMU0 on the SR5E1x.

Figure 51. CMU0 block diagram

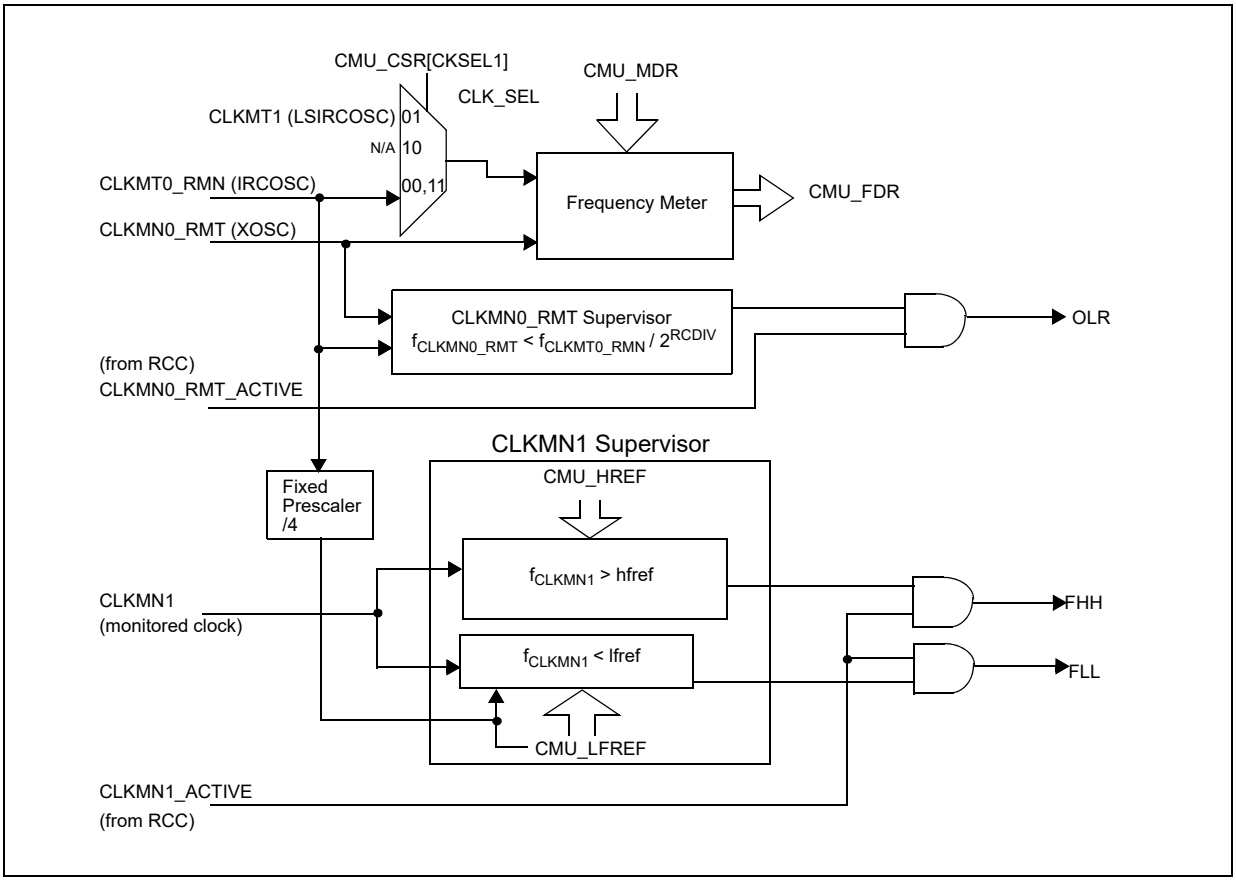
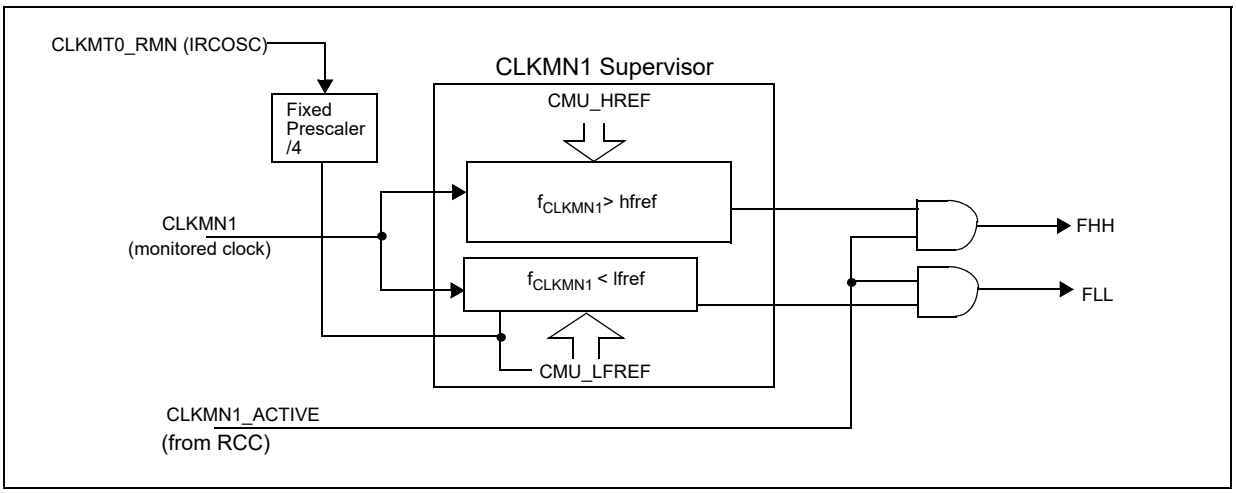


Figure 52 shows the block diagram for the other CMUs on the SR5E1x.

Figure 52. Other CMUs block diagram



11.3.1.1 Clock input sources

Table 113 shows the clocks that are monitored by each CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. IRCOSC is the reference clock for all clock monitors. Only CMU0 implements the XOSC

monitor. CMU0 uses the IRCOSC clock to measure if the XOSC is too low. CMU0 can also be used to calibrate the IRCOSC frequency using the XOSC. All other CMUs are configured identically.

Table 113. Clock input sources

Clock module	Monitored clock	CMU group
CMU_0_PLL0_XOSC_IRCOSC	PLL0:PHI, XOSC, IRCOSC, LSIRCOSC	—
CMU_1_CORE	SYS_CLK	CMU_Platform
CMU_2_APB1	APB1_CLK	CMU_Platform
CMU_3_APB2	APB2_CLK	CMU_Platform
CMU_4_SARADC	SARADC_CLK	CMU_other
CMU_5_SDADC	SDADC_CLK	CMU_other

Table 114. CMU clock signals

Clock signals	Functions	CMU_0	CMU_1	CMU_2	CMU_3	CMU_4	CMU_5
CLKMT1	Measured clock against CLKMN0_RMT.	LSIRCOSC	—				
CLKMN1	Monitored clock (CLKMT0_RMN as reference).	Refer to *Table 113: Clock input sources*.					
CLKMT0_RMN	– Reference clock for CLKMN1 monitoring. – Measured clock against CLKMN0_RMT.	IRCOSC					
CLKMN0_RMT	– Reference clock for CLKMT0_RMN measure. – Monitored clock, lower limit only (CLKMNT0_RMN as reference).	XOSC	—				

11.3.1.2 CMU registers and field availability

Table 115 specifies which registers and fields are available for a given CMU.

Table 115. CMU register availability

Address offset	Register	CMU	Note
0x0000	CMU_CSR	All	CMU_CSR[SFM] and CMU_CSR[CKSEL1] in CMU0 only.
0x0004	CMU_FDR	CMU0	—
0x0008	CMU_HFREFR	All	—
0x000C	CMU_LFREFR	All	—
0x0010	CMU_ISR	All	CMU_ISR[OLRI] in CMU0 only.
0x0014	Reserved	—	—
0x0018	CMU_MDR	CMU0	—

11.3.1.3 CMU register write protection

The CMU registers defined in *Table 116* have write protection with:

- Soft locking: can be unlocked by software after being previously locked.
- Hard locking: can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the CMU register write protection (refer to *Section 5.8.6: REG_PROT configuration* in *Chapter 5: Device configuration* for details).

Table 116. CMU register write protection

Offset	Register(1)
0x0000	CMU Control Status Register (CMU_CSR)
0x0004	CMU Frequency Display Register (CMU_FDR)
0x0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)
0x000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)
0x0010	CMU Interrupt Status Register (CMU_ISR)
0x0014	Reserved
0x0018	CMU Measurement Duration Register (CMU_MDR)

1. Refer to Register Protection chapter for bit field details.

11.3.2 PLL0 monitor

Software can program an upper and lower limit on the expected PLL0 PHI output clock frequency. If the monitor is enabled and the measured frequency is above or below the limits, a flag bit is set and an interrupt, if enabled, is generated. The default condition of the clock monitor is disabled.

11.3.3 External oscillator (XOSC) monitor

The XOSC frequency is compared with a minimum value limit. If the measured XOSC frequency is below the limit, a flag is set and an interrupt, if enabled, is generated.

11.3.4 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU0, using the XOSC as a reference. This allows for application trimming of the IRCOSC frequency.

11.3.5 System clock monitors

A CMU is assigned to monitor the frequency of the peripheral bridges, peripheral, and ADC clocks (refer to *Figure 47*).

11.3.6 Progressive system clock switching

In order to prevent sudden voltage drops and overshoots due to frequency and load changes, the RCC can ramp the system clock frequency down and/or up based on the power level values of the current and target modes.

During ramp-down, the rate of the frequency change is based on the RCC_PCS_SDUR, RCC_PCS_DIVC*n*, and RCC_PCS_DIVE*n* registers, where *n* corresponds to the current system clock source selection.

During ramp-up, the rate of the frequency change is based on the RCC_PCS_SDUR, RCC_PCS_DIVC*n*, and RCC_PCS_DIVS*n* registers, where n corresponds to the target system clock source selection.

11.3.6.1 Generic clock change requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in *Equation 1:* .

Equation 1

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

11.3.6.2 Configuration of RCC_PCS_SDUR

The switch duration field RCC_PCS_SDUR[SDUR] defines the duration of one system clock switch step in terms of 16 MHz int. RC osc.. After the expiration of this time, the module changes the clock divider value which changes the frequency of the system clock.

11.3.6.3 Configuration of RCC_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the PCS_DIVC*n*[RATE] register is given in *Table 117*.

Table 117. RCC_PCS_DIVC*n*[RATE] values

a_{max}	d	RCC_PCS_DIVCn[RATE]
0.05	0.012	12
0.10	0.048	48
0.15	0.112	112
0.20	0.184	184

11.3.6.4 Generic clock change properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{div} from the source clock with frequency f_{src} is given in *Equation 2:* .

Equation 2

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16MHz}\right)}{d}} \right\rceil$$

11.3.6.5 Clock ramp-down

The clock ramp-down starts with the divider value 1 and with the given divider increment value PCD_DIVCn[RATE] and ends with the given divider value PCS_DIVEn[DIVE].

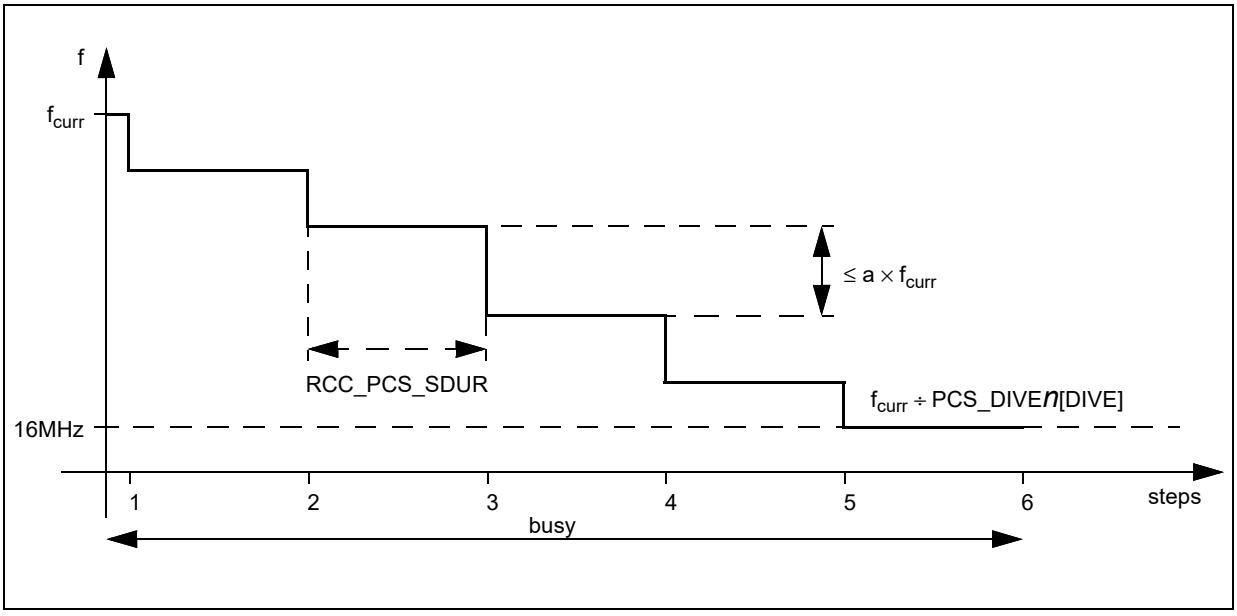
The divider end value for clock ramp-down is given in *Equation 3:* .

Equation 3

$$\textbf{PCS_DIVEn[DIVE]} = \frac{f_{src}}{16MHz} \times 1000 - 1$$

Where f_{curr} is the frequency of the currently selected system clock source.

Figure 53. System clock ramp-down timing (k = 6 example)



11.3.6.6 Clock ramp-up

The clock ramp-up starts with the given divider value PCS_DIVSn[DIVS] and with the given divider decrement value PCD_DIVCn[INIT] and ends with the divider value 1.

The initial divider change start value PCS_DIVCn[INIT] for system clock ramp-up is given in *Equation 4:* .

Equation 4

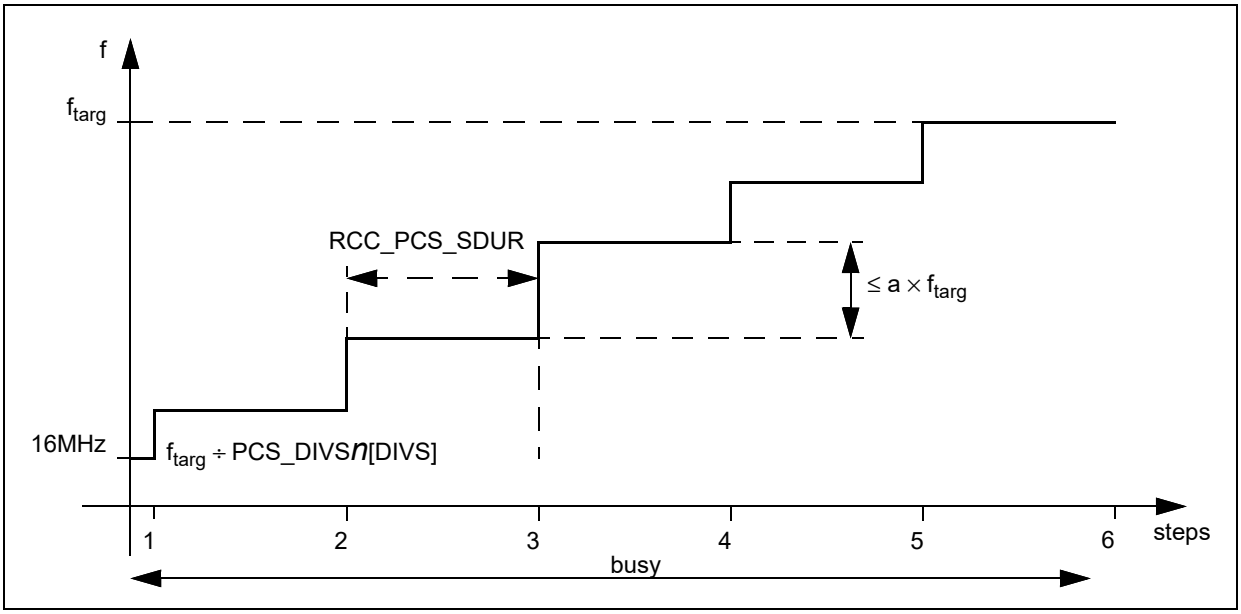
$$\textbf{PCS_DIVCn[INIT]} = d \times k \times 1000 - 1$$

Where k is calculated from *Equation 2:* using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in *Equation 5:* .

Equation 5

$$\textbf{PCS_DIVSn[DIVS]} = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

Figure 54. System clock ramp-up timing (k = 6 example)



11.4 Low-power modes

11.4.1 Peripheral clock gating

The clocks of unused AHB and APB peripherals can be disabled by software via the AHBxxENR and APBxxENR registers of the RCC.

By default after reset all peripherals clocks are disabled.

11.4.2 Core sleep mode (CSleep)

CSleep mode stops the CPU clock. Each core can enter CSleep independently by executing Wait For Interrupt/Event (WFI, WFE) instruction.

To further reduce power consumption in CSleep mode, each core can disable the clocks of the peripherals it is handling (and therefore may not be active when the core is in CSleep).

Each core has a set of Corex APB/AHB Sleep Mode Enable registers (Cx_AHBxxSMENR, Cx_APBxxSMENR) that allows the clock gating of the peripherals that are disabled together with the core, when CSleep mode is entered.

Each set of RCC registers can be written only by the associated core (that is C1_AxxxSMENR registers can only be written by Core1).

The status of the clock of a peripheral is determined by the corresponding bit settings in the registers that control the clock in RCC, as summarized in *Table 118: IP Clock status basing on the corresponding bits in RCC Clock enable registers*.

Table 118. IP Clock status basing on the corresponding bits in RCC Clock enable registers

Core 1	Core 2	AxxxENR[IPn]	C1_AxxSMENR [IPn]	C2_AxxxSMEN [IPn]	IPn Clock
ON	ON	0	-	-	OFF
		1	-	-	ON
SLEEP	ON	1	0	-	OFF
		1	1	-	ON
ON	SLEEP	1	-	0	OFF
		1	-	1	ON
SLEEP	SLEEP	1	0	-	OFF
		1	-	0	OFF
		1	1	1	ON

Note: From application point of view, one peripheral has to be controlled only by one core.

For example, only one of the two bits C1_AHBxxSMENR[UART1] and C2_AHBxxSMENR[UART1] has to be at 0, to put the peripheral in sleep mode.

In case both C1_AHBxxSMENR[UART1] and C2_AHBxxSMENR[UART1] are at 0 (meaning peripheral clock must be stopped), the peripheral stops as soon as one of the two cores goes in sleep.

11.5 RCC registers

Table 119. RCC memory map

Offset	Register Name	Register Title	Reset
0x0	*CR*	Clock Sources Control Register	0x0000 0000
0x4	*ICSR*	Internal Clock Sources Calibration Register	0x0000 0000
0x8	*CFGR*	Clock Configuration Register	0x0000 0024
0xC	*PLLCFGR*	PLL Configuration Register	0x0000 0000
0x18	*CIER*	Clock Interrupt / Fault Enable	0x0000 0000
0x1C	*CIFR*	Clock Interrupt / Fault Flag	0x0000 0000
0x20	*CICR*	Clock Interrupt / Fault Clear	0x0000 0000
0x24	*CCIPR1*	Peripherals independent clock configuration (selection) register 1	0x0000 0000
0x28	*CCIPR2*	Peripherals independent clock configuration (prescaler) register 2	0x2431 8842
0x2C	*LSCFGR*	Low Speed Clock Configuration register	0x0000 0000
0x30	*CRRCR*	Clock recovery RC register	0x0000 0000
0x50	*AHB1LRSTR*	AHB1 peripherals resets [Low Word]	0x0000 0000
0x54	*AHB1HRSTR*	AHB1 peripherals resets [High Word]	0x0000 0000
0x58	*AHB2LRSTR*	AHB2 peripherals resets [Low Word]	0x0000 0000
0x5C	*AHB2HRSTR*	AHB2 peripherals resets [High Word]	0x0000 0000
0x60	*APB1LRSTR*	APB1 peripherals resets [Low Word]	0x0000 0000
0x68	*APB2LRSTR*	APB2 peripherals resets [Low Word]	0x0000 0000
0x6C	*APB2HRSTR*	APB2 peripherals resets [High Word]	0x0000 0000
0x70	*AHB1LENR*	AHB1 peripherals clock enables [Low Word]	0x0000 0000
0x74	*AHB1HENR*	AHB1 peripherals clock enables [High Word]	0x0000 0000
0x78	*AHB2LENR*	AHB2 peripherals clock enables [Low Word]	0x0000 0000
0x7C	*AHB2HENR*	AHB2 peripherals clock enables [High Word]	0x0000 0000
0x80	*APB1LENR*	APB1 peripherals clock enables [Low Word]	0x0000 0030
0x88	*APB2LENR*	APB2 peripherals clock enables [Low Word]	0x0000 0000
0x8C	*APB2HENR*	APB2 peripherals clock enables [High Word]	0x0000 0000
0x90	*C1_AHB1LSMENR*	Core 1 AHB1 peripherals sleep mode clock enables [Low Word]	0x0003 3800
0x94	*C1_AHB1HSMENR*	Core 1 AHB1 peripherals sleep mode clock enables [High Word]	0x3000 1061
0x98	*C1_AHB2LSMENR*	Core 1 AHB2 peripherals sleep mode clock enables [Low Word]	0x0003 01FF
0x9C	*C1_AHB2HSMENR*	Core 1 AHB2 peripherals sleep mode clock enables [High Word]	0x0000 1071

Table 119. RCC memory map (continued)

Offset	Register Name	Register Title	Reset
0xA0	*C1_APB1LSMENR*	Core 1 APB1 peripherals sleep mode clock enables [Low Word]	0x19EF 8433
0xA8	*C1_APB2LSMENR*	Core 1 APB2 peripherals sleep mode clock enables [Low Word]	0x0987 9805
0xAC	*C1_APB2HSMENR*	Core 1 APB2 peripherals sleep mode clock enables [High Word]	0x0001 5503
0xB0	*C2_AHB1LSMENR*	Core 2 AHB1 peripherals sleep mode clock enables [Low Word]	0x0003 3800
0xB4	*C2_AHB1HSMENR*	Core 2 AHB1 peripherals sleep mode clock enables [High Word]	0x3000 1061
0xB8	*C2_AHB2LSMENR*	Core 2 AHB2 peripherals sleep mode clock enables [Low Word]	0x0003 01FF
0xBC	*C2_AHB2HSMENR*	Core 2 AHB2 peripherals sleep mode clock enables [High Word]	0x0000 1071
0xC0	*C2_APB1LSMENR*	Core 2 APB1 peripherals sleep mode clock enables [Low Word]	0x19EF 8433
0xC8	*C2_APB2LSMENR*	Core 2 APB2 peripherals sleep mode clock enables [Low Word]	0x0987 9805
0xCC	*C2_APB2HSMENR*	Core 2 APB2 peripherals sleep mode clock enables [High Word]	0x0001 5503
0xD0	*DBGCR*	Debug Clock Control Register	0x0005 033F
0xD4	*CMUR*	Clock Monitor Unit Control Register	0x0000 0300
0xD8	*HSM_SW_CTRLR*	HSM SW Control Register	0x0000 0000
0x100	*DSWRR*	Destructive Software Reset Register	0x0000 0000
0x104	*DRETR*	Destructive Reset Escalation Threshold Register	0x0000 000F
0x108	*DESR*	Destructive Event Status Register	0x0000 0000
0x10C	*DORER*	Destructive Output Reset Enable Register	0x0101 C73B
0x200	*FSWRR*	Functional Software Reset Register	0x0000 0000
0x204	*FRETR*	Functional Reset Escalation Threshold Register	0x0000 000F
0x208	*FESR*	Functional Event Status Register	0x0000 0000
0x20C	*FORER*	Functional Output Reset Enable Register	0x0000 0004
0x210	*FERDR*	Functional Event Reset Disable Register	0x0000 0000
0x214	*FESSR*	Functional Event Short Sequence Register	0x0000 0040
0x300	*SYS_CFG_REG*	Global System Configuration Register	0x0000 0001
0x304	*C1_VTOR_INIT_REG*	Core1 Vector Table Offset Init Register	0x1FF0 0100
0x308	*C2_VTOR_INIT_REG*	Core2 Vector Table Offset Init Register	0x0000 0000
0x30C	*C1_VTOR_ADDR_REG*	Core1 User VTOR Address Register	0x0000 0000
0x310	*C1_BOOT_CTRL_REG*	Core1 Boot Control Register	0x0000 0003

Table 119. RCC memory map (continued)

Offset	Register Name	Register Title	Reset
0x314	*C2_BOOT_CTRL_REG*	Core2 Boot Control Register	0x0000 0000
0x318	*C_SLEEP_STS_REG*	Cores Sleep Status register	0x0000 0000
0x380	*PCS_CTRL_REG*	PCS Control Register	0x0000 0002
0x384	*PCS_SDUR_REG*	PCS Switch Duration Register	0x0000 0000
0x388	*PCS_DIVC_XOSC_REG*	PCS Divider Change XOSC Register	0x03E7 0000
0x38C	*PCS_DIVE_XOSC_REG*	PCS Divider End XOSC Register	0x0000 03E7
0x390	*PCS_DIVS_XOSC_REG*	PCS Divider Start XOSC Register	0x0000 03E7
0x394	*PCS_DIVC_PLL0_REG*	PCS Divider Change PLL0 Register	0x03E7 0000
0x398	*PCS_DIVE_PLL0_REG*	PCS Divider End PLL0 Register	0x0000 03E7
0x39C	*PCS_DIVS_PLL0_REG*	PCS Divider Start PLL0 Register	0x0000 03E7
0x3A0	*PCS_DIVC_PLL1_REG*	PCS Divider Change PLL1 Register	0x03E7 0000
0x3A4	*PCS_DIVE_PLL1_REG*	PCS Divider End PLL1 Register	0x0000 03E7
0x3A8	*PCS_DIVS_PLL1_REG*	PCS Divider Start PLL1 Register	0x0000 03E7

11.6 RCC register descriptions

CR Clock Sources Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		PLL1LOCK	PLL1ON	RESERVED		PLL0LOCK	PLL0ON	RESERVED					XOSCBYP	XOSCRDY	XOSCON	RESERVED					IRCOSCRDY	RESERVED								LSIRDY	LSION
R		R	RW	R		R	RW	R					RW	R	RW	R					R	R								R	RW

Address: RCC_REGMAPBaseAddress + 0x0

Type: RW

Reset: 0x0000 0000

Description: Clock Sources Control Register

Register reset: long functional reset (PHASE1 reset)

[29] **PLL1LOCK**: PLL1 Lock Status
- 0: PLL1 is unlocked
- 1: PLL1 is locked

[28] **PLL1ON**: PLL1 Enable (active HIGH). Write 0 allowed only if System PLL is not used as system clock

[25] **PLL0LOCK**: PLL0 Lock Status
- 0: PLL0 is unlocked
- 1: PLL0 is locked

[24] **PLL0ON**: PLL0 Enable (active HIGH). Write 0 allowed only if System PLL is not used as system clock

[18] **XOSCBYP**: High Speed External Clock Bypass
0: External crystal mode
1: External source mode (bypass)

[17] **XOSCRDY**: High Speed External Clock ready status
0: XOSC oscillator is not stable
1: XOSC oscillator is stable

[16] **XOSCON**: High Speed External Clock Enable (active HIGH). Write 0 allowed only if SYSCLK is not derived from XOSC

[10] **IRCOSCRDY**: High Speed Internal Clock ready status
0: IRCOSC oscillator is not stable
1: IRCOSC oscillator is stable

[1] **LSIRDY**: Low Speed Internal Clock ready status
0: LSI oscillator is not stable
1: LSI oscillator is stable

[0] **LSION**: Low Speed Internal Clock Enable (active HIGH)

ICSR Internal Clock Sources Calibration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		IRCOSC_TRIM_USER						IRCOSC_TRIM								RESERVED										IRCOSC_FT_CODE					
R		RW						R								R										R					

Address: RCC_REGMAPBaseAddress + 0x4

Type: RW

Reset: 0x0000 0000

Description: Internal clock sources calibration register. Refer to *Section 11.2.2: Internal RC oscillator (IRCOSC)*.

Register reset: power on reset (POR)

[29:24] **IRCOSC_TRIM_USER**: IRCOSC software trimming bits

[23:16] **IRCOSC_TRIM**: IRCOSC trimming bits from DCF

[5:0] **IRCOSC_FT_CODE**: IRCOSC fine trimming code from parabolic interpolator

CFGR Clock Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				MCOPRE				MCOSEL				RESERVED			SYSPRE									RESERVED		SWS			SW		
R				RW				RW				R			RW									R		R			RW		

Address: RCC_REGMAPBaseAddress + 0x8

Type: RW

Reset: 0x0000 0024

Description: Clock Configuration Register

Register reset: short functional reset (PHASE3 reset)

[29:24] **MCOPRE**: Monitor Clock Output Prescaler. Div Factor = MCOPRE x2
0x0: no division
0x1: Div Factor = 2
0x2: Div Factor = 4
...
0x3F: Div Factor = 126
Reset value: 0x0

[23:20] **MCOSEL**: Monitor Clock Output Selector
Note: MCOSEL must be updated only when MCOPRE = 0
0x8: LSI
0x9: IRCOSC
0xA: XOSC
0xB: PLL0
0xC: PLL1
Reset value: 0x0

[16:8] **SYSPRE**: System Clock Prescaler. Div Factor = SYSPRE+1
0x0: no division
0x1: Div Factor = 2
0x2: Div Factor = 3
...
0x1FF: Div Factor = 512
Reset value: 0x0

[5:3] **SWS**: System Clock Switch Status. Report the actual System Clock source. Refer to SW bit field.
Reset value: 0x4

[2:0] **SW**: System Clock Switch
A write access to this bit field loads the PCS Control. Refer to PCS_CTRL_REG
0x4: internal 16 MHz (IRCOSC) oscillator selected as system clock
0x5: external 4-40 MHz (XOSC) oscillator selected as system clock
0x6: PLL0 selected as system clock
0x7: PLL1 selected as system clock
others: not allowed (a write operation on SW bit field has effect only if SW[2]=1)
Reset value: 0x4

PLLCFGR PLL Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						PLL1SRC		RESERVED						PLL0SRC	
R																						RW		R						RW	

Address: RCC_REGMAPBaseAddress + 0xC

Type: RW

Reset: 0x0000 0000

Description: PLL Configuration Register

Register reset: long functional reset (PHASE1 reset)

[9:8] **PLL1SRC**: PLL1 Source Clock Selector. Write allowed only if PLL1 is disabled
0x0: None
0x1: None
0x2: PLL0:PHI
0x3: XOSC

[1:0] **PLL0SRC**: PLL0 Source Clock Selector. Write allowed only if PLL0 is disabled
0x0: None
0x1: None
0x2: IRCOSC
0x3: XOSC

CIER Clock Interrupt / Fault Enable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															SAFEMODE_FE	RESERVED									PLL1LOCK_IE	PLL0LOCK_IE	XOSCRDY_IE	RESERVED			LSIRDY_IE
R															RW	R									RW	RW	RW	R			RW

Address: RCC_REGMAPBaseAddress + 0x18

Type: RW

Reset: 0x0000 0000

Description: Clock Interrupt / Fault Enable (active HIGH)

Register reset: short functional reset (PHASE3 reset)

[16] **SAFEMODE_FE**: SafeMode FCCU Fault Enable

[6] **PLL1LOCK_IE**: PLL1 Lock Interrupt Enable

[5] **PLL0LOCK_IE**: PLL0 Lock Interrupt Enable

[4] **XOSCRDY_IE**: XOSC Ready Interrupt Enable

[0] **LSIRDY_IE**: LSI Ready Interrupt Enable

CIFR Clock Interrupt / Fault Flag

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															SAFEMODE_FF	RESERVED									PLL1LOCK_IF	PLL0LOCK_IF	XOSCRDY_IF	RESERVED			LSIRDY_IF
R															R	R									R	R	R	R			R

Address: RCC_REGMAPBaseAddress + 0x1C

Type: R

Reset: 0x0000 0000

Description: Clock Interrupt / Fault Flag (active HIGH)

Register reset: short functional reset (PHASE3 reset)

[16] **SAFEMODE_FF**: SafeMode Fault Flag. This flag is set by hardware when System Clock source is switched to IRCOSC

[6] **PLL1LOCK_IF**: PLL1 Lock Interrupt Flag. This flag is set by hardware when PLL1 is locked.

[5] **PLL0LOCK_IF**: PLL0 Lock Interrupt Flag. This flag is set by hardware when PLL0 is locked.

[4] **XOSCRDY_IF**: XOSC Ready Interrupt Flag. This flag is set by hardware when XOSC oscillator is stable.

[0] **LSIRDY_IF**: LSI Ready Interrupt Flag. This flag is set by hardware when LSI oscillator is stable.

CICR Clock Interrupt / Fault Clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															SAFEMODE_FC	RESERVED									PLL1LOCK_IC	PLL0LOCK_IC	XOSCRDY_IC	RESERVED			LSIRDY_IC
R															W	R									W	W	W	R			W

Address: RCC_REGMAPBaseAddress + 0x20

Type: W

Reset: 0x0000 0000

Description: Clock Interrupt / Fault Clear

Register reset: short functional reset (PHASE3 reset)

[16] **SAFEMODE_FC**: Safemode Fault Clear.
Writing 1 to this bit clears the relative CIFR bit
Reset by HW

[6] **PLL1LOCK_IC**: PLL1 Lock Interrupt Clear.
Writing 1 to this bit clears the relative CIFR bit
Reset by HW

[5] **PLL0LOCK_IC**: PLL0 Lock Interrupt Clear.
Writing 1 to this bit clears the relative CIFR bit
Reset by HW

[4] **XOSCRDY_IC**: XOSC Ready Interrupt Clear.
Writing 1 to this bit clears the relative CIFR bit
Reset by HW

[0] **LSIRDY_IC**: LSI Ready Interrupt Clear.
Writing 1 to this bit clears the relative CIFR bit
Reset by HW

CCIPR1 Peripherals independent clock configuration (selection) register 1

31 30 29 28 27 26 25 24 23	22 21 20	19	18 17 16	15	14 13 12	11	10 9 8	7	6 5 4	3	2 1 0
RESERVED	SD_ADC_SEL	RESERVED	ADC_SEL	RESERVED	FDCAN_SEL	RESERVED	I2S23_SEL	RESERVED	I2C_SEL	RESERVED	UART_SEL
R	RW	R	RW	R	RW	R	RW	R	RW	R	RW

Address: RCC_REGMAPBaseAddress + 0x24

Type: RW

Reset: 0x0000 0000

Description: Register reset: short functional reset (PHASE3 reset)

[22:20] **SD_ADC_SEL**: Sigma Delta ADC Clock Source selection
0x4: XOSC
0x5: PLL0
0x6: PLL1
others: None

[18:16] **ADC_SEL**: SAR ADC Clock Source selection
0x4: XOSC
0x5: PLL0
0x6: PLL1
others: None

[14:12] **FDCAN_SEL**: FDCAN Clock Source selection
0x4: XOSC
0x5: PLL0
others: None

[10:8] **I2S23_SEL**: I2S23 Clock Source selection
0x4: IRCOSC
0x5: XOSC
0x6: PLL0
0x7: I2S Pad
others: None

[6:4] **I2C_SEL**: I2C Clock Source selection
0x4: IRCOSC
0x5: XOSC
0x6: PLL0
others: None

[2:0] **UART_SEL**: UART Clock Source selection
0x4: IRCOSC
0x5: XOSC
0x6: PLL0
0x7: LSI
others: None

CCIPR2 Peripherals independent clock configuration (prescaler) register 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD_ADC_PRE						ADC_PRE						FDCAN_PRE					I2S23_PRE					I2C_PRE					UART_PRE			
RW						RW						RW					RW					RW					RW			

Address: RCC_REGMAPBaseAddress + 0x28

Type: RW

Reset: 0x2431 8842

Description: Register reset: short functional reset (PHASE3 reset)

[31:26] **SD_ADC_PRE**: Sigma Delta ADC Clock Prescaler. Div Factor = PRE x 2
0x0: no division
0x1: Div Factor = 2
0x2: Div Factor = 4
...
0x3F: Div Factor = 126
Reset value: 0x9

[25:20] **ADC_PRE**: SAR ADC Clock Prescaler. Div Factor = PRE x 2
0x0: no division
0x1: Div Factor = 2
0x2: Div Factor = 4
...
0x3F: Div Factor = 126
Reset value: 0x3

[19:15] **FDCAN_PRE**: FDCAN Clock Prescaler. Div Factor = PRE + 1
0x0: no division
0x1: Div Factor = 2
0x2: Div Factor = 3
...
0x1F: Div Factor = 32
Reset value: 0x3

[14:10] **I2S23_PRE**: I2S23 Clock Prescaler. Div Factor = PRE + 1
0x0: no division
0x1: Div Factor = 2
0x2: Div Factor = 3
...
0x1F: Div Factor = 32
Reset value: 0x2

[9:5] **I2C_PRE**: I2C Clock Prescaler. Div Factor = PRE + 1
0x0: no division
0x1: Div Factor = 2
0x2: Div Factor = 3
...
0x1F: Div Factor = 32
Reset value: 0x2

[4:0] **UART_PRE**: UART Clock Prescaler. Div Factor = PRE + 1
0x0: no division
0x1: Div Factor = 2
0x2: Div Factor = 3
...
0x1F: Div Factor = 32
Reset value: 0x2

LSCFGR Low Speed Clock Configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					LSCOSEL		LSCOEN	RESERVED													RTCSEL		RTCSTS	RESERVED			LSIPRE				
R					RW		RW	R													RW		R	R			RW				

Address: RCC_REGMAPBaseAddress + 0x2C

Type: RW

Reset: 0x0000 0000

Description: Register reset: short functional reset (PHASE3 reset)

[26:25] **LSCOSEL**: Low Speed Clock Output Monitor Source selection
0x0: None
0x1: None
0x2: LSI_DIVn : LSI after LSIPRE prescaler division
0x3: XOSC divided by 32

[24] **LSCOEN**: Low Speed Clock Output Monitor clock enable (active HIGH)

[10:9] **RTCSEL**: Real Time Clock Source selection
0x0: None
0x1: None
0x2: LSI_DIVn : LSI after LSIPRE prescaler division
0x3: XOSC divided by 32

[8] **RTCSTS**: Real Time Clock Status: if 1, RTC kernel clock is active and the relative reset is released

[4:0] **LSIPRE**: Low Speed Internal Oscillator prescaler. Div Factor = PRE x 2
0x0: no division
0x1: Div Factor = 2
0x2: Div Factor = 4
...
0x1F: Div Factor = 62

CRRCR Clock recovery RC register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															SYSCLK_FOR_SAFE_EN	RESERVED						PLL1_LOL_SAFE_EN	PLL0_LOL_SAFE_EN	RESERVED							XOSC_LOC_SAFE_EN
R															RW	R						RW	RW	R							RW

Address: RCC_REGMAPBaseAddress + 0x30

Type: RW

Reset: 0x0000 0000

Description: Register reset: short functional reset (PHASE3 reset)

[16] **SYSCLK_FOR_SAFE_EN**: SYSCLK Out of Range SafeMode Enable
1: switch SYSCLK to RCOSC if a SYSCLK is out of defined range

[9] **PLL1_LOL_SAFE_EN**: PLL1 Loss Of Lock SafeMode Enable
1: switch SYSCLK to RCOSC if SYSCLK is derived from PLL1 and a PLL1 LOL occurs

[8] **PLL0_LOL_SAFE_EN**: PLL0 Loss Of Lock SafeMode Enable
1: switch SYSCLK to RCOSC if SYSCLK is derived from PLL0 and a PLL0 LOL occurs

[0] **XOSC_LOC_SAFE_EN**: XOSC Loss Of Clock SafeMode Enable
1: switch SYSCLK to RCOSC if SYSCLK is derived from XOSC and a XOSC LOC occurs

AHB1LRSTR AHB1 peripherals resets [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															DMA2	RESERVED		CORDIC	CRC	XOSCM	RESERVED										
R															RW	R		RW	RW	RW	R										

Address: RCC_REGMAPBaseAddress + 0x50

Type: RW

Reset: 0x0000 0000

Description: AHB1 peripherals resets [Low Word].
0: does not reset
1: reset the relative peripheral

Register reset: short functional reset (PHASE3 reset)

[16] **DMA2**: DMA2 reset

[13] **CORDIC**: CORDIC reset

[12] **CRC**: CRC reset

[11] **HSEM**: HSEM reset

AHB1HRSTR AHB1 peripherals resets [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		HRTIMER2	HRTIMER1	RESERVED															COMP2_DIG	RESERVED					DAC4	DAC3	RESERVED				ADC3_ADC4_ADC5
R		RW	RW	R															RW	R					RW	RW	R				RW

Address: RCC_REGMAPBaseAddress + 0x54

Type: RW

Reset: 0x0000 0000

Description: AHB1 peripherals resets [High Word].
0: does not reset
1: reset the relative peripheral

Register reset: short functional reset (PHASE3 reset)

[29] **HRTIMER2**: HRTIMER2 reset

[28] **HRTIMER1**: HRTIMER1 reset

[12] **COMP2_DIG**: Digital 2 reset

[6] **DAC4**: DAC4 reset

[5] **DAC3**: DAC3 reset

[0] **ADC3_ADC4_ADC5**: SAR ADC345 reset

AHB2LRSTR AHB2 peripherals resets [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														DMAMUX1	DMA1	RESERVED							GPIOI	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
R														RW	RW	R							RW	RW	RW	RW	RW	RW	RW	RW	RW

Address: RCC_REGMAPBaseAddress + 0x58

Type: RW

Reset: 0x0000 0000

Description: AHB2 peripherals resets [Low Word].
0: does not reset
1: reset the relative peripheral

Register reset: short functional reset (PHASE3 reset)

[17] **DMAMUX1**: DMAMUX1 reset

[16] **DMA1**: DMA1 reset

[8] **GPIOI**: GPIOI reset

[7] **GPIOH**: GPIOH reset

[6] **GPIOG**: GPIOG reset

[5] **GPIOF**: GPIOF reset

[4] **GPIOE**: GPIOE reset

[3] **GPIOD**: GPIOD reset

[2] **GPIOC**: GPIOC reset

[1] **GPIOB**: GPIOB reset

[0] **GPIOA**: GPIOA reset

AHB2HRSTR AHB2 peripherals resets [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																			COMP1_DIG	RESERVED					DAC2	DAC1	BDAC1	RESERVED			ADC1_ADC2
R																			RW	R					RW	RW	RW	R			RW

Address: RCC_REGMAPBaseAddress + 0x5C

Type: RW

Reset: 0x0000 0000

Description: AHB2 peripherals resets [High Word].
0: does not reset
1: reset the relative peripheral

Register reset: short functional reset (PHASE3 reset)

[12] **COMP1_DIG**: COMP1 reset

[6] **DAC2**: DAC2 reset

[5] **DAC1**: DAC1 reset

[4] **BDAC1**: BDAC1 reset

[0] **ADC1_ADC2**: ADC12 reset

APB1LRSTR APB1 peripherals resets [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			UART3	UART2	RESERVED		SPI3	SPI2	I2C2	I2C1	RESERVED	TIM_TS	TIM7	TIM6	TIM3	TIM2	RESERVED				RTC	RESERVED				IWDG2	IWDG1	RESERVED		WWDG2	WWDG1
R			RW	RW	R		RW	RW	RW	RW	R	RW	RW	RW	RW	RW	R				RW	R				R	R	R		R	R

Address: RCC_REGMAPBaseAddress + 0x60

Type: RW

Reset: 0x0000 0000

Description: APB1 peripherals resets [Low Word].
0: does not reset
1: reset the relative peripheral

Register reset: short functional reset (PHASE3 reset)

[28] **UART3**: UART3 reset

[27] **UART2**: UART2 reset

[24] **SPI3**: SPI3 reset

[23] **SPI2**: SPI2 reset

[22] **I2C2**: I2C2 reset

[21] **I2C1**: I2C1 reset

[19] **TIM_TS**: TIM_TS reset

[18] **TIM7**: TIM7 reset

[17] **TIM6**: TIM6 reset

[16] **TIM3**: TIM3 reset

[15] **TIM2**: TIM2 reset

[10] **RTC**: RTC reset

[5] **IWDG2**: IWDG2 reset

[4] **IWDG1**: IWDG1 reset

[1] **WWDG2**: WWDG2 reset

[0] **WWDG1**: WWDG1 reset

APB2LRSTR APB2 peripherals resets [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				UART1	RESERVED		SPI4	SPI1	RESERVED				TIM16	TIM15	TIM5	TIM4	RESERVED		TIM8_PWM	TIM1_PWM	RESERVED								SMPU	RESERVED	SYSCFG
R				RW	R		RW	RW	R				RW	RW	RW	RW	R		RW	RW	R								RW	R	RW

Address: RCC_REGMAPBaseAddress + 0x68

Type: RW

Reset: 0x0000 0000

Description: APB2 peripherals resets [Low Word].
0: does not reset
1: reset the relative peripheral

Register reset: short functional reset (PHASE3 reset)

[27] **UART1**: UART1 reset

[24] **SPI4**: SPI4 reset

[23] **SPI1**: SPI1 reset

[18] **TIM16**: TIM16 reset

[17] **TIM15**: TIM15 reset

[16] **TIM5**: TIM5 reset

[15] **TIM4**: TIM4 reset

[12] **TIM8_PWM**: TIM8_PWM reset

[11] **TIM1_PWM**: TIM1_PWM reset

[2] **SMPU**: SMPU reset

[0] **SYSCFG**: SYSCFG reset

APB2HRSTR APB2 peripherals resets [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															FDCAN_MRAM	RESERVED	FDCAN4	RESERVED	FDCAN3	RESERVED	FDCAN2	RESERVED	FDCAN1	RESERVED							SD_ADC2	SD_ADC1
R															RW	R	RW	R	RW	R	RW	R	RW	R							RW	RW

Address: RCC_REGMAPBaseAddress + 0x6C

Type: RW

Reset: 0x0000 0000

Description: APB2 peripherals resets [High Word].
0: does not reset
1: reset the relative peripheral

Register reset: short functional reset (PHASE3 reset)

[16] **FDCAN_MRAM**: FDCAN_MRAM reset

[14] **FDCAN4**: FDCAN4 reset
Set by HW if APB2HRSTR[FDCAN_MRAM] = 1
Can be cleared only writing 0 also to APB2HRSTR[FDCAN_MRAM]

[12] **FDCAN3**: FDCAN3 reset
Set by HW if APB2HRSTR[FDCAN_MRAM] = 1
Can be cleared only writing 0 also to APB2HRSTR[FDCAN_MRAM]

[10] **FDCAN2**: FDCAN2 reset
Set by HW if APB2HRSTR[FDCAN_MRAM] = 1
Can be cleared only writing 0 also to APB2HRSTR[FDCAN_MRAM]

[8] **FDCAN1**: FDCAN1 reset
Set by HW if APB2HRSTR[FDCAN_MRAM] = 1
Can be cleared only writing 0 also to APB2HRSTR[FDCAN_MRAM]

[1] **SD_ADC2**: SD_ADC2 reset

[0] **SD_ADC1**: SD_ADC1 reset

AHB1LENR AHB1 peripherals clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															DMA2	RESERVED		CORDIC	CRC	HSEM	RESERVED											
R															RW	R		RW	RW	RW	R											

Address: RCC_REGMAPBaseAddress + 0x70

Type: RW

Reset: 0x0000 0000

Description: AHB1 peripherals clock enables [Low Word].
0: clock disabled
1: clock enabled

[16] **DMA2**: DMA2 clock enable

[13] **CORDIC**: CORDIC clock enable

[12] **CRC**: CRC clock enable

[11] **HSEM**: HSEM clock enable

AHB1HENR AHB1 peripherals clock enables [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		HRTIMER2	HRTIMER1	RESERVED															COMP2_DIG	RESERVED					DAC4	DAC3	RESERVED				ADC3_ADC4_ADC5
R		RW	RW	R															RW	R					RW	RW	R				RW

Address: RCC_REGMAPBaseAddress + 0x74

Type: RW

Reset: 0x0000 0000

Description: AHB1 peripherals clock enables [High Word].
0: clock disabled
1: clock enabled

Register reset: short functional reset (PHASE3 reset)

[29] **HRTIMER2**: HRTIMER2 clock enable

[28] **HRTIMER1**: HRTIMER1 clock enable

[12] **COMP2_DIG**: Digital Comparetor 2 clock enable

[6] **DAC4**: DAC4 clock enable

[5] **DAC3**: DAC3 clock enable

[0] **ADC3_ADC4_ADC5**: SAR ADC345 clock enable

AHB2LENR AHB2 peripherals clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														DMAMUX1	DMA1	RESERVED							GPIOI	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
R														RW	RW	R							RW	RW	RW	RW	RW	RW	RW	RW	RW

Address: RCC_REGMAPBaseAddress + 0x78

Type: RW

Reset: 0x0000 0000

Description: AHB2 peripherals clock enables [Low Word].
0: clock disabled
1: clock enabled

Register reset: short functional reset (PHASE3 reset)

[17] **DMAMUX1**: DMAMUX1 clock enable

[16] **DMA1**: DMA1 clock enable

[8] **GPIOI**: GPIOI clock enable

[7] **GPIOH**: GPIOH clock enable

[6] **GPIOG**: GPIOG clock enable

[5] **GPIOF**: GPIOF clock enable

[4] **GPIOE**: GPIOE clock enable

[3] **GPIOD**: GPIOD clock enable

[2] **GPIOC**: GPIOC clock enable

[1] **GPIOB**: GPIOB clock enable

[0] **GPIOA**: GPIOA clock enable

AHB2HENR AHB2 peripherals clock enables [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																			COMP1_DIG	RESERVED					DAC2	DAC1	BDAC1	RESERVED			ADC1_ADC2
R																			RW	R					RW	RW	RW	R			RW

Address: RCC_REGMAPBaseAddress + 0x7C

Type: RW

Reset: 0x0000 0000

Description: AHB2 peripherals clock enables [High Word].
0: clock disabled
1: clock enabled

Register reset: short functional reset (PHASE3 reset)

[12] **COMP1_DIG**: COMP1 clock enable

[6] **DAC2**: DAC2 clock enable

[5] **DAC1**: DAC1 clock enable

[4] **BDAC1**: BDAC1 clock enable

[0] **ADC1_ADC2**: ADC12 clock enable

APB1LENR APB1 peripherals clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			UART3	UART2	RESERVED		SPI3	SPI2	I2C2	I2C1	RESERVED	TIM_TS	TIM7	TIM6	TIM3	TIM2	RESERVED				RTC	RESERVED					IWDG2	IWDG1	RESERVED		WWDG2	WWDG1
R			RW	RW	R		RW	RW	RW	RW	R	RW	RW	RW	RW	RW	R				RW	R					R	R	R		RW1S	RW1S

Address: RCC_REGMAPBaseAddress + 0x80

Type: RW

Reset: 0x0000 0030

Description: APB1 peripherals clock enables [Low Word].
0: clock disabled
1: clock enabled

Register reset: short functional reset (PHASE3 reset)

[28] **UART3**: UART3 clock enable

[27] **UART2**: UART2 clock enable

[24] **SPI3**: SPI3 clock enable

[23] **SPI2**: SPI2 clock enable

[22] **I2C2**: I2C2 clock enable

[21] **I2C1**: I2C1 clock enable

[19] **TIM_TS**: TIM_TS clock enable

[18] **TIM7**: TIM7 clock enable

[17] **TIM6**: TIM6 clock enable

[16] **TIM3**: TIM3 clock enable

[15] **TIM2**: TIM2 clock enable

[10] **RTC**: RTC clock enable

[5] **IWDG2**: IWDG2 clock enable
Reset value: 0x1

[4] **IWDG1**: IWDG1 clock enable
Reset value: 0x1

[1] **WWDG2**: WWDG2 clock enable
Note: This bit is set by SW and cleared by HW.

[0] **WWDG1**: WWDG1 clock enable
Note: This bit is set by SW and cleared by HW.

APB2LENR APB2 peripherals clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				UART1	RESERVED		SPI4	SPI1	RESERVED				TIM16	TIM15	TIM5	TIM4	RESERVED		TIM8_PWM	TIM1_PWM	RESERVED								SMPU	RESERVED	SYSCFG
R				RW	R		RW	RW	R				RW	RW	RW	RW	R		RW	RW	R								RW	R	RW

Address: RCC_REGMAPBaseAddress + 0x88

Type: RW

Reset: 0x0000 0000

Description: APB2 peripherals clock enables [Low Word].
0: clock disabled
1: clock enabled

Register reset: short functional reset (PHASE3 reset)

[27] **UART1**: UART1 clock enable

[24] **SPI4**: SPI4 clock enable

[23] **SPI1**: SPI1 clock enable

[18] **TIM16**: TIM16 clock enable

[17] **TIM15**: TIM15 clock enable

[16] **TIM5**: TIM5 clock enable

[15] **TIM4**: TIM4 clock enable

[12] **TIM8_PWM**: TIM8_PWM clock enable

[11] **TIM1_PWM**: TIM1_PWM clock enable

[2] **SMPU**: SMPU clock enable

[0] **SYSCFG**: SYSCFG clock enable

APB2HENR APB2 peripherals clock enables [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															FDCAN_MRAM	RESERVED	FDCAN4	RESERVED	FDCAN3	RESERVED	FDCAN2	RESERVED	FDCAN1	RESERVED							SD_ADC2	SD_ADC1
R															R	R	RW	R	RW	R	RW	R	RW	R							RW	RW

Address: RCC_REGMAPBaseAddress + 0x8C

Type: RW

Reset: 0x0000 0000

Description: APB2 peripherals clock enables [High Word].
0: clock disabled
1: clock enabled

Register reset: short functional reset (PHASE3 reset)

[16] **FDCAN_MRAM**: FDCAN_MRAM clock enable
Set by HW if at least one the FDCANx_EN = 1

[14] **FDCAN4**: FDCAN4 clock enable

[12] **FDCAN3**: FDCAN3 clock enable

[10] **FDCAN2**: FDCAN2 clock enable

[8] **FDCAN1**: FDCAN1 clock enable

[1] **SD_ADC2**: SD_ADC2 clock enable

[0] **SD_ADC1**: SD_ADC1 clock enable

C1_AHB1LSMENR Core 1 AHB1 peripherals sleep mode clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															DMA2	RESERVED		CORDIC	CRC	HSEM	RESERVED										
R															RW	R		RW	RW	RW	R										

Address: RCC_REGMAPBaseAddress + 0x90

Type: RW

Reset: 0x0003 3800

Description: Core 1 AHB1 peripherals sleep mode clock enables [Low Word].
0: clock disabled, when Core 1 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C2_AxxxSMENR is 1, when Core 1 goes in Sleep mode. Write access allowed only for Core 1

Register reset: short functional reset (PHASE3 reset)

[16] **DMA2**: DMA2 sleep mode clock enable
Reset value: 0x1

[13] **CORDIC**: CORDIC sleep mode clock enable
Reset value: 0x1

[12] **CRC**: CRC sleep mode clock enable
Reset value: 0x1

[11] **HSEM**: HSEM sleep mode clock enable
Reset value: 0x1

C1_AHB1HSMENR Core 1 AHB1 peripherals sleep mode clock enables [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		HRTIMER2	HRTIMER1	RESERVED															COMP2_DIG	RESERVED					DAC4	DAC3	RESERVED				ADC3_ADC4_ADC5
R		RW	RW	R															RW	R					RW	RW	R				RW

Address: RCC_REGMAPBaseAddress + 0x94

Type: RW

Reset: 0x3000 1061

Description: Core 1 AHB1 peripherals sleep mode clock enables [High Word].
0: clock disabled, when Core 1 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C2_AxxxSMENR is 1, when Core 1 goes in Sleep mode. Write access allowed only for Core 1

Register reset: short functional reset (PHASE3 reset)

[29] **HRTIMER2**: HRTIMER2 sleep mode clock enable
Reset value: 0x1

[28] **HRTIMER1**: HRTIMER1 sleep mode clock enable
Reset value: 0x1

[12] **COMP2_DIG**: Digital Comparetor 2 sleep mode clock enable
Reset value: 0x1

[6] **DAC4**: DAC4 sleep mode clock enable
Reset value: 0x1

[5] **DAC3**: DAC3 sleep mode clock enable
Reset value: 0x1

[0] **ADC3_ADC4_ADC5**: SAR ADC345 sleep mode clock enable
Reset value: 0x1

C1_AHB2LSMENR Core 1 AHB2 peripherals sleep mode clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														DMAMUX1	DMA1	RESERVED							GPIOI	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
R														RW	RW	R							RW	RW	RW	RW	RW	RW	RW	RW	RW

Address: RCC_REGMAPBaseAddress + 0x98

Type: RW

Reset: 0x0003 01FF

Description: Core 1 AHB2 peripherals sleep mode clock enables1 [Low Word].
0: clock disabled, when Core 1 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C2_AxxxSMENR is 1, when Core 1 goes in Sleep mode. Write access allowed only for Core 1

Register reset: short functional reset (PHASE3 reset)

[17] **DMAMUX1**: DMAMUX1 sleep mode clock enable
Reset value: 0x1

[16] **DMA1**: DMA1 sleep mode clock enable
Reset value: 0x1

[8] **GPIOI**: GPIOI sleep mode clock enable
Reset value: 0x1

[7] **GPIOH**: GPIOH sleep mode clock enable
Reset value: 0x1

[6] **GPIOG**: GPIOG sleep mode clock enable
Reset value: 0x1

[5] **GPIOF**: GPIOF sleep mode clock enable
Reset value: 0x1

[4] **GPIOE**: GPIOE sleep mode clock enable
Reset value: 0x1

[3] **GPIOD**: GPIOD sleep mode clock enable
Reset value: 0x1

[2] **GPIOC**: GPIOC sleep mode clock enable
Reset value: 0x1

[1] **GPIOB**: GPIOB sleep mode clock enable
Reset value: 0x1

[0] **GPIOA**: GPIOA sleep mode clock enable
Reset value: 0x1

C1_AHB2HSMENR Core 1 AHB2 peripherals sleep mode clock enables [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																			COMP1_DIG	RESERVED					DAC2	DAC1	BDAC1	RESERVED			ADC1_ADC2
R																			RW	R					RW	RW	RW	R			RW

Address: RCC_REGMAPBaseAddress + 0x9C

Type: RW

Reset: 0x0000 1071

Description: Core 1 AHB2 peripherals sleep mode clock enables [High Word].
0: clock disabled, when Core 1 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C2_AxxxSMENR is 1, when Core 1 goes in Sleep mode. Write access allowed only for Core 1

Register reset: short functional reset (PHASE3 reset)

[12] **COMP1_DIG**: COMP1 sleep mode clock enable
Reset value: 0x1

[6] **DAC2**: DAC2 sleep mode clock enable
Reset value: 0x1

[5] **DAC1**: DAC1 sleep mode clock enable
Reset value: 0x1

[4] **BDAC1**: BDAC1 sleep mode clock enable
Reset value: 0x1

[0] **ADC1_ADC2**: ADC12 sleep mode clock enable
Reset value: 0x1

C1_APB1LSMENR Core 1 APB1 peripherals sleep mode clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			UART3	UART2	RESERVED		SPI3	SPI2	I2C2	I2C1	RESERVED	TIM_TS	TIM7	TIM6	TIM3	TIM2	RESERVED				RTC	RESERVED				IWDG2	IWDG1	RESERVED		WWDG2	WWDG1
R			RW	RW	R		RW	RW	RW	RW	R	RW	RW	RW	RW	RW	R				RW	R				RW	RW	R		R	R

Address: RCC_REGMAPBaseAddress + 0xA0

Type: RW

Reset: 0x19EF 8433

Description: Core 1 APB1 peripherals sleep mode clock enables [Low Word].
0: clock disabled, when Core 1 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C2_AxxxSMENR is 1, when Core 1 goes in Sleep mode. Write access allowed only for Core 1

Register reset: short functional reset (PHASE3 reset)

[28] **UART3**: UART3 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[27] **UART2**: UART2 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[24] **SPI3**: SPI3 sleep mode clock enable
Reset value: 0x1

[23] **SPI2**: SPI2 sleep mode clock enable
Reset value: 0x1

[22] **I2C2**: I2C2 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[21] **I2C1**: I2C1 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[19] **TIM_TS**: TIM_TS sleep mode clock enable
Reset value: 0x1

[18] **TIM7**: TIM7 sleep mode clock enable
Reset value: 0x1

[17] **TIM6**: TIM6 sleep mode clock enable
Reset value: 0x1

[16] **TIM3**: TIM3 sleep mode clock enable
Reset value: 0x1

[15] **TIM2**: TIM2 sleep mode clock enable
Reset value: 0x1

[10] **RTC**: RTC sleep mode clock enable
Reset value: 0x1

[5] **IWDG2**: IWDG2 sleep mode clock enable
Reset value: 0x1

[4] **IWDG1**: IWDG1 sleep mode clock enable
Reset value: 0x1

[1] **WWDG2**: WWDG2 sleep mode clock enable
Reset value: 0x1

[0] **WWDG1**: WWDG1 sleep mode clock enable
Reset value: 0x1

C1_APB2LSMENR Core 1 APB2 peripherals sleep mode clock enables [Low Word]

31 30 29 28	27	26 25	24	23	22 21 20 19	18	17	16	15	14 13	12	11	10 9 8 7 6 5 4 3	2	1	0
RESERVED	UART1	RESERVED	SPI4	SPI1	RESERVED	TIM16	TIM15	TIM5	TIM4	RESERVED	TIM8_PWM	TIM1_PWM	RESERVED	SMPU	RESERVED	SYSCFG
R	RW	R	RW	RW	R	RW	RW	RW	RW	R	RW	RW	R	RW	R	RW

Address: RCC_REGMAPBaseAddress + 0xA8

Type: RW

Reset: 0x0987 9805

Description: Core 1 APB2 peripherals sleep mode clock enables [Low Word].
0: clock disabled, when Core 1 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C2_AxxxSMENR is 1, when Core 1 goes in Sleep mode. Write access allowed only for Core 1

Register reset: short functional reset (PHASE3 reset)

[27] **UART1**: UART1 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[24] **SPI4**: SPI4 sleep mode clock enable
Reset value: 0x1

[23] **SPI1**: SPI1 sleep mode clock enable
Reset value: 0x1

[18] **TIM16**: TIM16 sleep mode clock enable
Reset value: 0x1

[17] **TIM15**: TIM15 sleep mode clock enable
Reset value: 0x1

[16] **TIM5**: TIM5 sleep mode clock enable
Reset value: 0x1

[15] **TIM4**: TIM4 sleep mode clock enable
Reset value: 0x1

[12] **TIM8_PWM**: TIM8_PWM sleep mode clock enable
Reset value: 0x1

[11] **TIM1_PWM**: TIM1_PWM sleep mode clock enable
Reset value: 0x1

[2] **SMPU**: SMPU sleep mode clock enable
Reset value: 0x1

[0] **SYSCFG**: SYSCFG sleep mode clock enable
Reset value: 0x1

C1_APB2HSMENR Core 1 APB2 peripherals sleep mode clock enables [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															FDCAN_MRAM	RESERVED	FDCAN4	RESERVED	FDCAN3	RESERVED	FDCAN2	RESERVED	FDCAN1	RESERVED							SD_ADC2	SD_ADC1
R															R	R	RW	R	RW	R	RW	R	RW	R							RW	RW

Address: RCC_REGMAPBaseAddress + 0xAC

Type: RW

Reset: 0x0001 5503

Description: Core 1 APB2 peripherals sleep mode clock enables [High Word].
0: clock disabled, when Core 1 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C2_AxxxSMENR is 1, when Core 1 goes in Sleep mode. Write access allowed only for Core 1

Register reset: short functional reset (PHASE3 reset)

[16] **FDCAN_MRAM**: FDCAN_MRAM sleep mode clock enable
Set by HW if at least one the FDCANx_SMEN = 1
Reset value: 0x1

[14] **FDCAN4**: FDCAN4 sleep mode clock enable
Reset value: 0x1

[12] **FDCAN3**: FDCAN3 sleep mode clock enable
Reset value: 0x1

[10] **FDCAN2**: FDCAN2 sleep mode clock enable
Reset value: 0x1

[8] **FDCAN1**: FDCAN1 sleep mode clock enable
Reset value: 0x1

[1] **SD_ADC2**: SD_ADC2 sleep mode clock enable
Reset value: 0x1

[0] **SD_ADC1**: SD_ADC1 sleep mode clock enable
Reset value: 0x1

C2_AHB1LSMENR Core 2 AHB1 peripherals sleep mode clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															DMA2	RESERVED		CORDIC	CRC	HSEM	RESERVED										
R															RW	R		RW	RW	RW	R										

Address: RCC_REGMAPBaseAddress + 0xB0

Type: RW

Reset: 0x0003 3800

Description: Core 2 AHB1 peripherals sleep mode clock enables [Low Word].
0: clock disabled, when Core 2 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C1_AxxxSMENR is 1, when Core 2 goes in Sleep mode. Write access allowed only for Core 2

Register reset: short functional reset (PHASE3 reset)

[16] **DMA2**: DMA2 sleep mode clock enable
Reset value: 0x1

[13] **CORDIC**: CORDIC sleep mode clock enable
Reset value: 0x1

[12] **CRC**: CRC sleep mode clock enable
Reset value: 0x1

[11] **HSEM**: HSEM sleep mode clock enable
Reset value: 0x1

C2_AHB1HSMENR Core 2 AHB1 peripherals sleep mode clock enables [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		HRTIMER2	HRTIMER1	RESERVED															COMP2_DIG	RESERVED					DAC4	DAC3	RESERVED				ADC3_ADC4_ADC5
R		RW	RW	R															RW	R					RW	RW	R				RW

Address: RCC_REGMAPBaseAddress + 0xB4

Type: RW

Reset: 0x3000 1061

Description: Core 2 AHB1 peripherals sleep mode clock enables [High Word].
0: clock disabled
1: clock enabled if the relative bit of registers AxxxENR and C1_AxxxSMENR is 1
Write access allowed only for Core 2

Register reset: short functional reset (PHASE3 reset)

[29] **HRTIMER2**: HRTIMER2 sleep mode clock enable
Reset value: 0x1

[28] **HRTIMER1**: HRTIMER1 sleep mode clock enable
Reset value: 0x1

[12] **COMP2_DIG**: Digital Comparetor 2 sleep mode clock enable
Reset value: 0x1

[6] **DAC4**: DAC4 sleep mode clock enable
Reset value: 0x1

[5] **DAC3**: DAC3 sleep mode clock enable
Reset value: 0x1

[0] **ADC3_ADC4_ADC5**: SAR ADC345 sleep mode clock enable
Reset value: 0x1

C2_AHB2LSMENR Core 2 AHB2 peripherals sleep mode clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														DMAMUX1	DMA1	RESERVED							GPIOI	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
R														RW	RW	R							RW	RW	RW	RW	RW	RW	RW	RW	RW

Address: RCC_REGMAPBaseAddress + 0xB8

Type: RW

Reset: 0x0003 01FF

Description: Core 2 AHB2 peripherals sleep mode clock enables [Low Word].
0: clock disabled
1: clock enabled if the relative bit of registers AxxxENR and C1_AxxxSMENR is 1
Write access allowed only for Core 2

Register reset: short functional reset (PHASE3 reset)

[17] **DMAMUX1**: DMAMUX1 sleep mode clock enable
Reset value: 0x1

[16] **DMA1**: DMA1 sleep mode clock enable
Reset value: 0x1

[8] **GPIOI**: GPIOI sleep mode clock enable
Reset value: 0x1

[7] **GPIOH**: GPIOH sleep mode clock enable
Reset value: 0x1

[6] **GPIOG**: GPIOG sleep mode clock enable
Reset value: 0x1

[5] **GPIOF**: GPIOF sleep mode clock enable
Reset value: 0x1

[4] **GPIOE**: GPIOE sleep mode clock enable
Reset value: 0x1

[3] **GPIOD**: GPIOD sleep mode clock enable
Reset value: 0x1

[2] **GPIOC**: GPIOC sleep mode clock enable
Reset value: 0x1

[1] **GPIOB**: GPIOB sleep mode clock enable
Reset value: 0x1

[0] **GPIOA**: GPIOA sleep mode clock enable
Reset value: 0x1

C2_AHB2HSMENR Core 2 AHB2 peripherals sleep mode clock enables [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																			COMP1_DIG	RESERVED					DAC2	DAC1	BDAC1	RESERVED			ADC1_ADC2
R																			RW	R					RW	RW	RW	R			RW

Address: RCC_REGMAPBaseAddress + 0xBC

Type: RW

Reset: 0x0000 1071

Description: Core 2 AHB2 peripherals sleep mode clock enables [High Word].
0: clock disabled, when Core 2 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C1_AxxxSMENR is 1, when Core 2 goes in Sleep mode. Write access allowed only for Core 2

Register reset: short functional reset (PHASE3 reset)

[12] **COMP1_DIG**: COMP1 sleep mode clock enable
Reset value: 0x1

[6] **DAC2**: DAC2 sleep mode clock enable
Reset value: 0x1

[5] **DAC1**: DAC1 sleep mode clock enable
Reset value: 0x1

[4] **BDAC1**: BDAC1 sleep mode clock enable
Reset value: 0x1

[0] **ADC1_ADC2**: ADC12 sleep mode clock enable
Reset value: 0x1

C2_APB1LSMENR Core 2 APB1 peripherals sleep mode clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			UART3	UART2	RESERVED		SPI3	SPI2	I2C2	I2C1	RESERVED	TIM_TS	TIM7	TIM6	TIM3	TIM2	RESERVED				RTC	RESERVED				IWDG2	IWDG1	RESERVED		WWDG2	WWDG1
R			RW	RW	R		RW	RW	RW	RW	R	RW	RW	RW	RW	RW	R				RW	R				RW	RW	R		R	R

Address: RCC_REGMAPBaseAddress + 0xC0

Type: RW

Reset: 0x19EF 8433

Description: Core 2 APB1 peripherals sleep mode clock enables [Low Word].
0: clock disabled, when Core 2 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C1_AxxxSMENR is 1, when Core 2 goes in Sleep mode. Write access allowed only for Core 2

Register reset: short functional reset (PHASE3 reset)

[28] **UART3**: UART3 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[27] **UART2**: UART2 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[24] **SPI3**: SPI3 sleep mode clock enable
Reset value: 0x1

[23] **SPI2**: SPI2 sleep mode clock enable
Reset value: 0x1

[22] **I2C2**: I2C2 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[21] **I2C1**: I2C1 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[19] **TIM_TS**: TIM_TS sleep mode clock enable
Reset value: 0x1

[18] **TIM7**: TIM7 sleep mode clock enable
Reset value: 0x1

[17] **TIM6**: TIM6 sleep mode clock enable
Reset value: 0x1

[16] **TIM3**: TIM3 sleep mode clock enable
Reset value: 0x1

[15] **TIM2**: TIM2 sleep mode clock enable
Reset value: 0x1

[10] **RTC**: RTC sleep mode clock enable
Reset value: 0x1

[5] **IWDG2**: IWDG2 sleep mode clock enable
Reset value: 0x1

[4] **IWDG1**: IWDG1 sleep mode clock enable
Reset value: 0x1

[1] **WWDG2**: WWDG2 sleep mode clock enable
Reset value: 0x1

[0] **WWDG1**: WWDG1 sleep mode clock enable
Reset value: 0x1

C2_APB2LSMENR Core 2 APB2 peripherals sleep mode clock enables [Low Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				UART1	RESERVED		SPI4	SPI1	RESERVED				TIM16	TIM15	TIM5	TIM4	RESERVED		TIM8_PWM	TIM1_PWM	RESERVED								SMPU	RESERVED	SYSCFG
R				RW	R		RW	RW	R				RW	RW	RW	RW	R		RW	RW	R								RW	R	RW

Address: RCC_REGMAPBaseAddress + 0xC8

Type: RW

Reset: 0x0987 9805

Description: Core 2 APB2 peripherals sleep mode clock enables [Low Word].
0: clock disabled
1: clock enabled if the relative bit of registers AxxxENR and C1_AxxxSMENR is 1
Write access allowed only for Core 2

Register reset: short functional reset (PHASE3 reset)

[27] **UART1**: UART1 sleep mode clock enable
Note: If AxxxSMNER bit is 0, the IP can directly enable its clock in Sleep Mode (through a clock request) only if the relative AxxxENR bit is 1.
Reset value: 0x1

[24] **SPI4**: SPI4 sleep mode clock enable
Reset value: 0x1

[23] **SPI1**: SPI1 sleep mode clock enable
Reset value: 0x1

[18] **TIM16**: TIM16 sleep mode clock enable
Reset value: 0x1

[17] **TIM15**: TIM15 sleep mode clock enable
Reset value: 0x1

[16] **TIM5**: TIM5 sleep mode clock enable
Reset value: 0x1

[15] **TIM4**: TIM4 sleep mode clock enable
Reset value: 0x1

[12] **TIM8_PWM**: TIM8_PWM sleep mode clock enable
Reset value: 0x1

[11] **TIM1_PWM**: TIM1_PWM sleep mode clock enable
Reset value: 0x1

[2] **SMPU**: SMPU sleep mode clock enable
Reset value: 0x1

[0] **SYSCFG**: SYSCFG sleep mode clock enable
Reset value: 0x1

C2_APB2HSMENR Core 2 APB2 peripherals sleep mode clock enables [High Word]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															FDCAN_MRAM	RESERVED	FDCAN4	RESERVED	FDCAN3	RESERVED	FDCAN2	RESERVED	FDCAN1	RESERVED							SD_ADC2	SD_ADC1
R															R	R	RW	R	RW	R	RW	R	RW	R							RW	RW

Address: RCC_REGMAPBaseAddress + 0xCC

Type: RW

Reset: 0x0001 5503

Description: Core 2 APB2 peripherals sleep mode clock enables [High Word].
0: clock disabled, when Core 2 goes in Sleep mode
1: clock enabled if the relative bit of registers AxxxENR and C1_AxxxSMENR is 1, when Core 2 goes in Sleep mode. Write access allowed only for Core 2

Register reset: short functional reset (PHASE3 reset)

[16] **FDCAN_MRAM**: FDCAN_MRAM sleep mode clock enable
Set by HW if at least one the FDCANx_SMEN = 1
Reset value: 0x1

[14] **FDCAN4**: FDCAN4 sleep mode clock enable
Reset value: 0x1

[12] **FDCAN3**: FDCAN3 sleep mode clock enable
Reset value: 0x1

[10] **FDCAN2**: FDCAN2 sleep mode clock enable
Reset value: 0x1

[8] **FDCAN1**: FDCAN1 sleep mode clock enable
Reset value: 0x1

[1] **SD_ADC2**: SD_ADC2 sleep mode clock enable
Reset value: 0x1

[0] **SD_ADC1**: SD_ADC1 sleep mode clock enable
Reset value: 0x1

DBGCR Debug Clock Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							DBG_RST	RESERVED					ATB_TPIU_PRE			RESERVED						ATB_TPIU_CLK	ATB_CLK	RESERVED		DGB_CLK_5	DGB_CLK_4	DGB_CLK_3	DGB_CLK_2	DGB_CLK_1	DGB_CLK_0
R							RW	R					RW			R						RW	RW	R		RW	RW	RW	RW	RW	RW

Address: RCC_REGMAPBaseAddress + 0xD0

Type: RW

Reset: 0x0005 033F

Description: Register reset: power-on reset

[24] **DBG_RST**: Debug reset: If 1, nDBGPOR is asserted
Note: nDBGPOR can be asserted also by DAP_RESET_REQ.
nDBGPOR is released only if DBG_RST=0 and DAP_RESET_REQ=0
Reset value: 0x0

[18:16] **ATB_TPIU_PRE**: ATB TPIU prescaler. Div Factor = PRE + 1
0x0: no division
0x1: Div Factor = 2
...
0x7: Div Factor = 8
Reset value: 0x5

[9] **ATB_TPIU_CLK**: ATB TPIU clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

[8] **ATB_CLK**: ATB clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

[5] **DGB_CLK_5**: Debug 5 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

[4] **DGB_CLK_4**: Debug 4 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

[3] **DGB_CLK_3**: Debug 3 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

[2] **DGB_CLK_2**: Debug 2 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

[1] **DGB_CLK_1**: Debug 1 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

[0] **DGB_CLK_0**: Debug 0 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

CMUR Clock Monitor Unit Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																		CMU5EN	CMU4EN	CMU3EN	CMU2EN	CMU1EN	CMU0EN	RESERVED		CMU5RST	CMU4RST	CMU3RST	CMU2RST	CMU1RST	CMU0RST
R																		RW	RW	RW	RW	RW	RW	R		RW	RW	RW	RW	RW	RW

Address: RCC_REGMAPBaseAddress + 0xD4

Type: RW

Reset: 0x0000 0300

Description: Register reset: short functional reset (PHASE3 reset)

[13] **CMU5EN**: CMU5 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x0

[12] **CMU4EN**: CMU4 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x0

[11] **CMU3EN**: CMU3 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x0

[10] **CMU2EN**: CMU2 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x0

[9] **CMU1EN**: CMU1 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

[8] **CMU0EN**: CMU0 clock enable
0: Clock disabled
1: Clock enabled
Reset value: 0x1

[5] **CMU5RST**: CMU5 reset
0: Does not reset
1: Reset the relative peripheral
Reset value: 0x0

[4] **CMU4RST**: CMU4 reset
0: Does not reset
1: Reset the relative peripheral
Reset value: 0x0

[3] **CMU3RST**: CMU3 reset
0: Does not reset
1: Reset the relative peripheral
Reset value: 0x0

[2] **CMU2RST**: CMU2 reset
0: Does not reset
1: Reset the relative peripheral
Reset value: 0x0

[1] **CMU1RST**: CMU1 reset
0: Does not reset
1: Reset the relative peripheral
Reset value: 0x0

[0] **CMU0RST**: CMU0 reset
0: Does not reset
1: Reset the relative peripheral
Reset value: 0x0

HSM_SW_CTRLR HSM SW Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															HSM_SW_EN
R																															RW1S

Address: RCC_REGMAPBaseAddress + 0xD8

Type: RW

Reset: 0x0000 0000

Description: This register allows to activate HSM via SW. The setting of HSM_SW_EN bit releases reset and clock to HSM (that is to HSM secure core).

Register reset: short functional reset (PHASE3 reset)

[0] **HSM_SW_EN**: Reset release bit
0: HSM and secure core not activated by SW
1: HSM and secure core SW enable.
Note: This bit has effect only if HSM is not already enabled by DCF.
The writing of this bit is allowed under conditions reported in Security ADD.
Note: This bit is set by SW and cleared by HW.
Reset value: 0x0

DSWRR Destructive Software Reset Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSWR																															
W																															

Address: RCC_REGMAPBaseAddress + 0x100

Type: W

Reset: 0x0000 0000

Description: Writing 0x2EA055F8 to this register triggers a destructive reset

Register reset: destructive reset (PHASE0 reset)

[31:0] **DSWR**: Reset value: 0x0

DRETR Destructive Reset Escalation Threshold Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												DRET			
R																												RW			

Address: RCC_REGMAPBaseAddress + 0x104

Type: RW

Reset: 0x0000 000F

Description: This register sets the threshold for destructive reset escalation to keep the chip in the reset state until the next power-on reset triggers a new reset sequence.
Writing any value to this register resets the destructive reset counter

Register reset: power-on reset

[3:0] **DRET**: Destructive reset escalation threshold - If the value of this field is 0, the destructive reset escalation function is disabled. Any other value is the number of destructive resets which keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the DRET register is not written to beforehand.
Reset value: 0xf

DESR Destructive Event Status Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							D_VD	RESERVED							D_TSENS	D_HSM	D_SSCM	RESERVED			D_JTAG	D_SWT_Boot	D_EDR	RESERVED		D_STCU_CF	D_FCCU	D_RCC_SW	RESERVED	D_RESET_IN	D_POR
R							W 1C	R							W 1C	W 1C	W 1C	R			W 1C	W 1C	W 1C	R		W 1C	W 1C	W 1C	R	W 1C	W 1C

Address: RCC_REGMAPBaseAddress + 0x108

Type: RW

Reset: 0x0000 0000

Description: This register contains the status of the destructive reset sources. Register bits are cleared on write 1
0: No destructive reset event has occurred since either the last clear or the last power-on reset assertion
1: A destructive reset event has occurred

Register reset: power-on reset

[24] **D_VD**: Voltage out of range from LVDs and/or HVDs
Reset value: 0x0

[16] **D_TSENS**: Temperature sensor
Reset value: 0x0

[15] **D_HSM**: HSM destructive reset request
Reset value: 0x0

[14] **D_SSCM**: SSCM error during data reading
Reset value: 0x0

[10] **D_JTAG**: JTAG destructive reset request
Reset value: 0x0

[9] **D_SWT_Boot**: Boot watchdog timeout during NVM initialization
Reset value: 0x0

[8] **D_EDR**: Escalation destructive reset
Reset value: 0x0

[5] **D_STCU_CF**: STCU critical failure indicated by STCU_CF trigger
Reset value: 0x0

[4] **D_FCCU**: FOSU reset
Reset value: 0x0

[3] **D_RCC_SW**: Software destructive reset from RCC
Reset value: 0x0

[1] **D_RESET_IN**: RESET_IN pin
Reset value: 0x0

[0] **D_POR**: Power-On reset
Reset value: 0x0

DORER Destructive Output Reset Enable Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							D_VD	RESERVED							D_TSENS	D_HSM	D_SSCM	RESERVED			D_JTAG	D_SWT_Boot	D_EDR	RESERVED		D_STCU_CF	D_FCCU	D_RCC_SW	RESERVED	D_RESET_IN	D_POR
R							R	R							R	R	R	R			RW	R	R	R		R	R	R	R	R	R

Address: RCC_REGMAPBaseAddress + 0x10C

Type: RW

Reset: 0x0101 C73B

Description: This register enables the report of a destructive reset on the RESET_OUT pad.

Register reset: power-on reset

[24] **D_VD**: Voltage out of range from LVDs and/or HVDs
Reset value: 0x1

[16] **D_TSENS**: Temperature sensor
Reset value: 0x1

[15] **D_HSM**: HSM destructive reset request
Reset value: 0x1

[14] **D_SSCM**: SSCM error during data reading
Reset value: 0x1

[10] **D_JTAG**: JTAG destructive reset request
Reset value: 0x1

[9] **D_SWT_Boot**: Boot watchdog timeout during NVM initialization
Reset value: 0x1

[8] **D_EDR**: Escalation destructive reset (generated by functional reset escalation)
Reset value: 0x1

[5] **D_STCU_CF**: STCU critical failure indicated by STCU_CF trigger
Reset value: 0x1

[4] **D_FCCU**: FOSU reset
Reset value: 0x1

[3] **D_RCC_SW**: Software destructive reset from RCC
Reset value: 0x1

[1] **D_RESET_IN**: RESET_IN pin
Reset value: 0x1

[0] **D_POR**: Power-On reset
Reset value: 0x1

FSWRR Functional Software Reset Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSWR																															
W																															

Address: RCC_REGMAPBaseAddress + 0x200

Type: W

Reset: 0x0000 0000

Description: Writing 0xF41B84D1 to this register triggers a functional reset

Register reset: short functional reset (PHASE3 reset)

[31:0] **FSWR**: Reset value: 0x0

FRETR Functional Reset Escalation Threshold Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												FRET			
R																												RW			

Address: RCC_REGMAPBaseAddress + 0x204

Type: RW

Reset: 0x0000 000F

Description: This register sets the threshold for functional reset escalation to a destructive reset. Writing any value to this register resets the functional reset counter.

Register reset: destructive reset (PHASE0 reset)

[3:0] **FRET**: Functional reset escalation threshold - If the value of this field is 0, the functional reset escalation function is disabled. Any other value is the number of functional resets which cause a destructive reset (if func_rst_escalation is connected to one of the dest_rst inputs) if the FRET register is not written to beforehand.
Reset value: 0xf

FESR Functional Event Status Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							F_VD	RESERVED							F_TSENS	F_HSM	RESERVED				F_JTAG	RESERVED			F_FCCU_SHORT	F_FCCU_LONG	RESERVED	F_RCC_SW	F_ST_DONE	RESERVED	
R							W 1C	R							W 1C	W 1C	R				W 1C	R			W 1C	W 1C	R	W 1C	W 1C	R	

Address: RCC_REGMAPBaseAddress + 0x208

Type: R

Reset: 0x0000 0000

Description: This register contains the status of the functional reset sources. Register bits are cleared on write 1
0: No functional reset event has occurred since either the last clear or the last power-on reset assertion
1: A functional reset event has occurred

Register reset: power-on reset

[24] **F_VD**: Voltage detector for peripheral domain
Reset value: 0x0

[16] **F_TSENS**: Temperature sensor
Reset value: 0x0

[15] **F_HSM**: HSM functional reset request
Reset value: 0x0

[10] **F_JTAG**: JTAG functional reset request
Reset value: 0x0

[6] **F_FCCU_SHORT**: Short functional reset
Reset value: 0x0

[5] **F_FCCU_LONG**: Long functional reset
Reset value: 0x0

[3] **F_RCC_SW**: Software functional reset from RCC
Reset value: 0x0

[2] **F_ST_DONE**: Self-test functional reset
Reset value: 0x0

FORER Functional Output Reset Enable Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							F_VD	RESERVED							F_TSENS	F_HSM	RESERVED				F_JTAG	RESERVED			F_FCCU_SHORT	F_FCCU_LONG	RESERVED	F_RCC_SW	F_ST_DONE	RESERVED	
R							RW	R							RW	RW	R				RW	R			R	RW	R	RW	R	R	

Address: RCC_REGMAPBaseAddress + 0x20C

Type: RW

Reset: 0x0000 0004

Description: This register enables the report of a functional reset on the RESET_OUT pad.

Register reset: destructive reset (PHASE0 reset)

[24] **F_VD**: Voltage detector for peripheral domain
Reset value: 0x0

[16] **F_TSENS**: Temperature sensor
Reset value: 0x0

[15] **F_HSM**: HSM functional reset request
Reset value: 0x0

[10] **F_JTAG**: JTAG functional reset request
Reset value: 0x0

[6] **F_FCCU_SHORT**: Short functional reset
Reset value: 0x0

[5] **F_FCCU_LONG**: Long functional reset
Reset value: 0x0

[3] **F_RCC_SW**: Software functional reset from RCC
Reset value: 0x0

[2] **F_ST_DONE**: Self-test functional reset
Reset value: 0x1

FERDR Functional Event Reset Disable Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							F_VD	RESERVED							F_TSENS	F_HSM	RESERVED				F_JTAG	RESERVED			F_FCCU_SHORT	F_FCCU_LONG	RESERVED	F_RCC_SW	F_ST_DONE	RESERVED	
R							RW Once	R							RW Once	R	R				RW Once	R			R	R	R	RW Once	R	R	

Address: RCC_REGMAPBaseAddress + 0x210

Type: RWOnce

Reset: 0x0000 0000

Description: This register provides dedicated bits to disable functional reset sources.
0: functional reset event triggers a reset sequence
1: functional reset event is masked

Register reset: destructive reset (PHASE0 reset)

[24] **F_VD**: Voltage detector for peripheral domain
Reset value: 0x0

[16] **F_TSENS**: Temperature sensor
Reset value: 0x0

[15] **F_HSM**: HSM functional reset - always enabled
Reset value: 0x0

[10] **F_JTAG**: JTAG functional reset request
Reset value: 0x0

[6] **F_FCCU_SHORT**: Short functional reset - always enabled
Reset value: 0x0

[5] **F_FCCU_LONG**: Long functional reset - always enabled
Reset value: 0x0

[3] **F_RCC_SW**: Software functional reset from RCC
Reset value: 0x0

[2] **F_ST_DONE**: Self-test functional reset - always enabled
Reset value: 0x0

FESSR Functional Event Short Sequence Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							F_VD	RESERVED							F_TSENS	F_HSM	RESERVED				F_JTAG	RESERVED			F_FCCU_SHORT	F_FCCU_LONG	RESERVED	F_RCC_SW	F_ST_DONE	RESERVED	
R							RWOnce	R							RWOnce	RWOnce	R				RWOnce	R			R	R	R	RWOnce	R	R	

Address: RCC_REGMAPBaseAddress + 0x214

Type: RWOnce

Reset: 0x0000 0040

Description: This register defines if the functional reset event triggers a LONG or SHORT reset sequence.
0: LONG reset sequence starting from PHASE1
1: SHORT reset sequence starting from PHASE3

Register reset: destructive reset (PHASE0 reset)

[24] **F_VD**: Voltage detector for peripheral domain
Reset value: 0x0

[16] **F_TSENS**: Temperature sensor
Reset value: 0x0

[15] **F_HSM**: HSM functional reset request
Reset value: 0x0

[10] **F_JTAG**: JTAG functional reset request
Reset value: 0x0

[6] **F_FCCU_SHORT**: Short functional reset - it always triggers a short reset sequence
Reset value: 0x1

[5] **F_FCCU_LONG**: Long functional reset - it always triggers a long reset sequence
Reset value: 0x0

[3] **F_RCC_SW**: Software functional reset from RCC
Reset value: 0x0

[2] **F_ST_DONE**: Self-test functional reset - it always triggers a long reset sequence
Reset value: 0x0

SYS_CFG_REG Global System Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							OTA_EN	RESERVED							LS_DMA	RESERVED							LS_CORE1_2	RESERVED						BOOT_CORE2	BOOT_CORE1
R							R	R							R	R							R	R						R	R

Address: RCC_REGMAPBaseAddress + 0x300

Type: R

Reset: 0x0000 0001

Description: This read-only register mirrors the Boot Core configuration done by the corresponding Write Once DCF.
It reports the cores that have to be started by HW in parallel to the Boot CPU (Core1), that is always enabled.

Register reset: destructive reset (PHASE0 reset)

[24] **OTA_EN**: Enables OTA on flash
0: OTA disabled (default)
1: OTA enabled
Reset value: 0x0

[16] **LS_DMA**: Enables lockstep replicas for DMA1, DMA2 and DMAMUX
0: Lockstep for DMA1, DMA2 and DMAMUX disabled (default)
1: Lockstep for DMA1, DMA2 and DMAMUX enabled
Reset value: 0x0

[8] **LS_CORE1_2**: Enables lockstep configuration for Core1 and 2
Core2 has to be started together with Core1 (respecting lockstep delay constraints)
0: Lockstep for Core1 and Core2 disabled (default)
1: Lockstep for Core1 and Core2 enabled
Note: Only C1_VTOR dcf has to be considered as starting address
Reset value: 0x0

[1] **BOOT_CORE2**: Core2 enabled
0: Core2 is not enabled
1: Core2 is enabled
Reset value: 0x0

[0] **BOOT_CORE1**: Core1 enabled
1: Core1 enabled to start as Boot CPU
Reset value: 0x1

C1_VTOR_INIT_REG Core1 Vector Table Offset Init Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1_VTOR_INIT																									RESERVED						
RW																									R						

Address: RCC_REGMAPBaseAddress + 0x304

Type: RW

Reset: 0x1FF0 0100

Description: This register reports the VTOR address for Boot CPU, at reset release.
After reset it reports the address from which the Boot CPU has started fetching. It depends on device state and BAF configuration.
Reset Value: BAF start address
Value after short functional reset assertion:
- When the flash is virgin, VTOR of the Boot CPU has to be the starting address of SRAM1.
- When flash is not virgin:
-- if BAF (Boot Assist Flash) is enabled (default), the VTOR has to be set to starting address of the BAF (Boot Assist Flash);
-- if BAF is bypassed by DCF setting, the VTOR has to be taken from Alternate Core1 User Address DCF.(Write Once DCF).

Register reset: destructive reset (PHASE0 reset)

[31:7] **C1_VTOR_INIT**: After reset it reports the address from which the Boot CPU has started fetching. It depends on device state and BAF configuration.
Default Value: BAF start address
Reset Value: See *Section 6.3.12.1: Boot CPU reset vector*.

C2_VTOR_INIT_REG Core2 Vector Table Offset Init Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2_VTOR_INIT																									RESERVED				UPDATE_COUNTS		
RW																									R				R		

Address: RCC_REGMAPBaseAddress + 0x308

Type: RW

Reset: 0x0000 0000

Description: This register defines the VTOR_INIT address of Core2.
Reset Value: 0x0000_0000
Value after short functional reset assertion: Default Value or Value of Corex Vector Table Offset DCF, if programmed

Register reset: destructive reset (PHASE0 reset)

[31:7] **C2_VTOR_INIT**: Address of vector table of the M7 CORE2 after reset release

[2:0] **UPDATE_COUNTS**: Reports the number of times the DCF has been overwritten.
Max allowed is 4

C1_VTOR_ADDR_REG Core1 User Vector Table - Offset Init Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1_VTOR_ADDR																									RESERVED				UPDATE_COUNTS		
R																									R				R		

Address: RCC_REGMAPBaseAddress + 0x30C

Type: R

Reset: 0x0000 0000

Description: This read-only register reports the address to be used by BAF to jump to the application code.
Reset Value: 0x0000_0000

Register reset: destructive reset (PHASE0 reset)

[31:7] **C1_VTOR_ADDR**: Address of user code vector table.

[2:0] **UPDATE_COUNTS**: Reports the number of times the DCF has been overwritten.
Max allowed is 4

C1_BOOT_CTRL_REG Core1 Boot Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1_LOCK	RESERVED																													C1_CPU_WAIT_RELEASE	C1_RES_RELEASE
RW	R																													RW	RW

Address: RCC_REGMAPBaseAddress + 0x310

Type: RW

Reset: 0x0000 0003

Description: This register allows to control the behavior of Cores at reset release and allows the Boot CPU to wake up the other cores via SW in case no HW wake-up has been programmed.
Register reset: destructive reset (PHASE0 reset)

[31] **C1_LOCK**: Lock bit
0: Register Content is writable
1: Lock the register content till next reset
NOTE: Lock bit also locks the content of C1_VTOR_REG
Reset value: 0x0

[1] **C1_CPU_WAIT_RELEASE**: CPU_WAIT_RELEASE bit
0: CPU is in 'CPU WAIT' state
1: CPU WAIT is released
Reset value: 0x1

[0] **C1_RES_RELEASE**: Reset release bit
0: Corex is under reset
1: Corex reset is released
Reset value: 0x1

C2_BOOT_CTRL_REG Core2 Boot Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2_LOCK	RESERVED																													C2_CPU_WAIT_RELEASE	C2_RES_RELEASE
RW	R																													RW	RW

Address: RCC_REGMAPBaseAddress + 0x314

Type: RW

Reset: 0x0000 0000

Description: This register allows to control the behavior of Cores at reset release and allows the Boot CPU to wake up the other cores via SW in case no HW wake-up has been programmed.

Register reset: destructive reset (PHASE0 reset)

[31] **C2_LOCK**: Lock bit
0: Register Content is writable
1: Lock the register content till next reset
NOTE: Lock bit also locks the content of C2_VTOR_REG
Reset value: 0x0

[1] **C2_CPU_WAIT_RELEASE**: CPU_WAIT_RELEASE bit
0: CPU is in 'CPU WAIT' state
1: CPU WAIT is released
Reset value: 0x0

[0] **C2_RES_RELEASE**: Reset release bit
0: Corex is under reset
1: Corex reset is released
Reset value: 0x0

C_SLEEP_STS_REG Cores Sleep Status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							C2_SLEEP_MODE_STS	RESERVED							C1_SLEEP_MODE_STS
R																							R	R							R

Address: RCC_REGMAPBaseAddress + 0x318

Type: R

Reset: 0x0000 0000

Description: This register reports the Sleep Mode status of the Cores

[8] **C2_SLEEP_MODE_STS**: Core 2 Sleep Mode Status
1: Core is in Sleep Mode
Reset value: 0x0

[0] **C1_SLEEP_MODE_STS**: Core 1 Sleep Mode Status
1: Core is in Sleep Mode
Reset value: 0x0

PCS_CTRL_REG PCS Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							PCS_BUSY	RESERVED					PCS_CTL		
R																							R	R					RW		

Address: RCC_REGMAPBaseAddress + 0x380

Type: RW

Reset: 0x0000 0002

Description: This register allows to control the behavior of the progressive clock switching module.

Register reset: power-on reset

[8] **PCS_BUSY**: PCS busy status

[2:0] **PCS_CTL**: PCS control:
-000: Idle
-001: ramp-up (allowed only after ramp-down)
-010: clock switch
-011: clock switch followed by ramp-up
-100: ramp-down
-101: ramp-down followed by ramp-up
-110: ramp-down followed by clock switch
-111: ramp-down followed by clock switch followed by ramp-up
PCS controls load request is generated upon a write access to CFGR[SW] register.
If PCS_CTL[1]=1 (clock switch enabled) the PCS controls are loaded only if the selected source is different from the current source, that is CFGR[SW] written is different from CFGR[SWS].
Reset value: 0x2

PCS_SDUR_REG PCS Switch Duration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDUR								RESERVED																							
RW								R																							

Address: RCC_REGMAPBaseAddress + 0x384

Type: RW

Reset: 0x0000 0000

Description: This register contains the progressive system clock switching duration for each step.

Register reset: power-on reset

[31:24] **SDUR**: Switch duration
This value defines the duration of one PCS clock switch step in terms of IRCOSC clock cycles
Reset value: 0x0

PCS_DIVC_XOSC_REG PCS Divider Change XOSC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INIT																RESERVED								RATE							
RW																R								RW							

Address: RCC_REGMAPBaseAddress + 0x388

Type: RW

Reset: 0x03E7 0000

Description: These registers define the rate of frequency change and initial change value on frequency ramp-up for the progressive system clock switching when system clock source is XOSC.

Register reset: power-on reset

[31:16] **INIT**: Divider change initial value

This is initial change value of the clock divider for the clock ramp-up phase

Byte write accesses are not allowed. Such an access does not result in an exception (that is ips_xfr_err is not asserted), however the value is not loaded with the new value.

Reset value: 0x3e7

[7:0] **RATE**: Divider change rate

This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase.

Reset value: 0x0

PCS_DIVE_XOSC_REG PCS Divider End XOSC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												DIVE																			
R												RW																			

Address: RCC_REGMAPBaseAddress + 0x38C

Type: RW

Reset: 0x0000 03E7

Description: These registers define the final division value on frequency ramp-down for the progressive system clock switching when system clock source is XOSC.

Register reset: power-on reset

[19:0] **DIVE**: Divider end value

This is the clock divider end value for the clock ramp-down phase.

Byte and half-word write accesses are not allowed for this register. Such an access does not result in an exception (that is ips_xfr_err is not asserted), however the value is not loaded with the new value.

Reset value: 0x3e7

PCS_DIVS_XOSC_REG PCS Divider Start XOSC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												DIVS																			
R												RW																			

Address: RCC_REGMAPBaseAddress + 0x390

Type: RW

Reset: 0x0000 03E7

Description: These registers define the initial division value on frequency ramp-up for the progressive system clock switching when system clock source is XOSC.

Register reset: power-on reset

[19:0] **DIVS**: Divider start value

This is the start value of the clock divider for the clock ramp-up phase.

Byte and half-word write accesses are not allowed for this register. Such an access does not result in an exception (that is ips_xfr_err is not asserted), however the value is not loaded with the new value.

Reset value: 0x3e7

PCS_DIVC_PLL0_REG PCS Divider Change PLL0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INIT																RESERVED								RATE							
RW																R								RW							

Address: RCC_REGMAPBaseAddress + 0x394

Type: RW

Reset: 0x03E7 0000

Description: These registers define the rate of frequency change and initial change value on frequency ramp-up for the progressive system clock switching when system clock source is PLL0.

Register reset: power-on reset

[31:16] **INIT**: Divider change initial value

This is initial change value of the clock divider for the clock ramp-up phase

Byte write accesses are not allowed. Such an access does not result in an exception (that is ips_xfr_err is not asserted), however the value is not loaded with the new value.

Reset value: 0x3e7

[7:0] **RATE**: Divider change rate

This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase.

Reset value: 0x0

PCS_DIVE_PLL0_REG PCS Divider End PLL0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												DIVE																			
R												RW																			

Address: RCC_REGMAPBaseAddress + 0x398

Type: RW

Reset: 0x0000 03E7

Description: These registers define the final division value on frequency ramp-down for the progressive system clock switching when system clock source is PLL0.

Register reset: power-on reset

[19:0] **DIVE**: Divider End Value

This is the clock divider end value for the clock ramp-down phase.

Byte and half-word write accesses are not allowed for this register. Such an access does not result in an exception (that is ips_xfr_err is not asserted), however the value is not loaded with the new value.

Reset value: 0x3e7

PCS_DIVS_PLL0_REG PCS Divider Start PLL0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												DIVS																			
R												RW																			

Address: RCC_REGMAPBaseAddress + 0x39C

Type: RW

Reset: 0x0000 03E7

Description: These registers define the initial division value on frequency ramp-up for the progressive system clock switching when system clock source is PLL0.

Register reset: power-on reset

[19:0] **DIVS**: Divider Start Value

This is the start value of the clock divider for the clock ramp-up phase.

Byte and half-word write accesses are not allowed for this register. Such an access does not result in an exception (that is ips_xfr_err is not asserted), however the value is not loaded with the new value.

Reset value: 0x3e7

PCS_DIVC_PLL1_REG PCS Divider Change PLL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INIT																RESERVED								RATE							
RW																R								RW							

Address: RCC_REGMAPBaseAddress + 0x3A0

Type: RW

Reset: 0x03E7 0000

Description: These registers define the rate of frequency change and initial change value on frequency ramp-up for the progressive system clock switching when system clock source is PLL1.

Register reset: power-on reset

[31:16] **INIT**: Divider Change Initial Value

This is initial change value of the clock divider for the clock ramp-up phase

Byte write accesses are not allowed. Such an access does not result in an exception (that is ips_xfr_err is not asserted), however the value is not loaded with the new value.

Reset value: 0x3e7

[7:0] **RATE**: Divider Change Rate

This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase.

Reset value: 0x0

PCS_DIVE_PLL1_REG PCS Divider End PLL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												DIVE																			
R												RW																			

Address: RCC_REGMAPBaseAddress + 0x3A4

Type: RW

Reset: 0x0000 03E7

Description: These registers define the final division value on frequency ramp-down for the progressive system clock switching when system clock source is PLL1.

Register reset: power-on reset

[19:0] **DIVE**: Divider End Value

This is the clock divider end value for the clock ramp-down phase.

Byte and half-word write accesses are not allowed for this register. Such an access does not result in an exception (that is ips_xfr_err is not asserted), however the value is not loaded with the new value.

Reset value: 0x3e7

PCS_DIVS_PLL1_REG PCS Divider Start PLL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												DIVS																			
R												RW																			

Address: RCC_REGMAPBaseAddress + 0x3A8

Type: RW

Reset: 0x0000 03E7

Description: These registers define the initial division value on frequency ramp-up for the progressive system clock switching when system clock source is PLL1.

Register reset: power-on reset

[19:0] **DIVS**: Divider Start Value

This is the start value of the clock divider for the clock ramp-up phase.

Byte and half-word write accesses are not allowed for this register. Such an access does not result in an exception (that is ips_xfr_err is not asserted), however the value is not loaded with the new value.

Reset value: 0x3e7

12 Dual PLL digital interface (PLLDIG)

12.1 Introduction

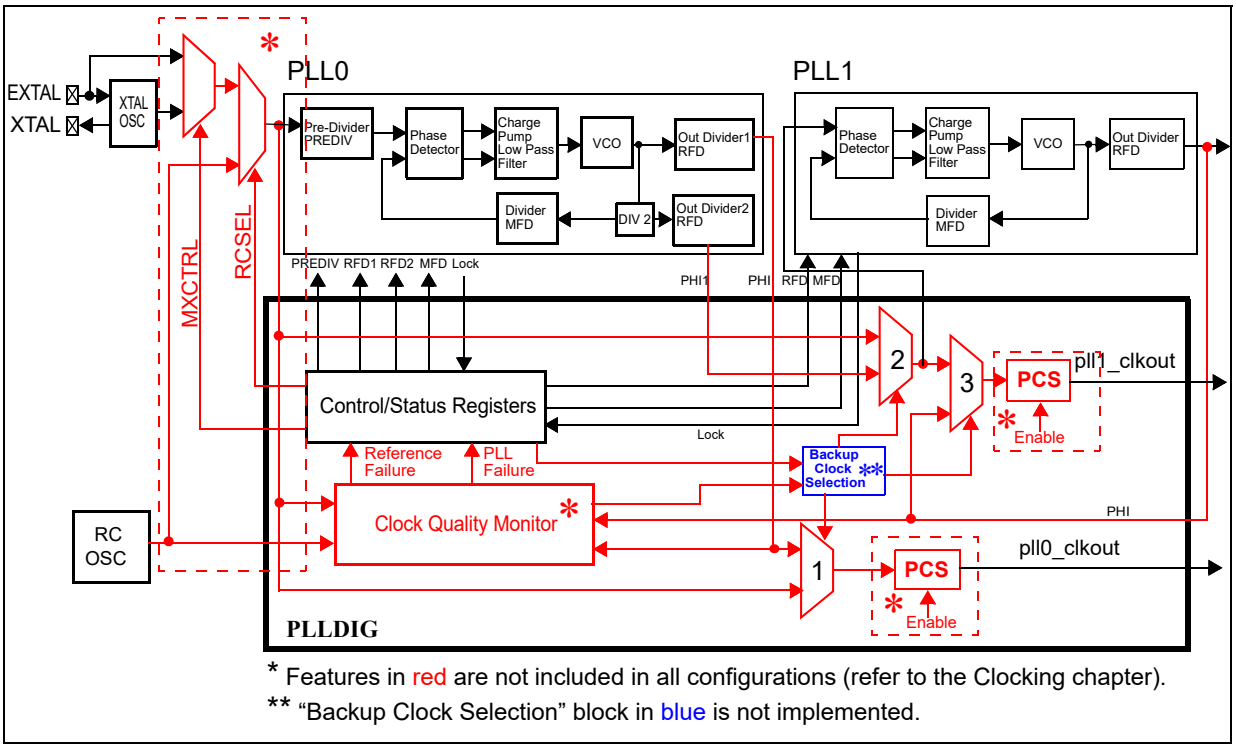
The MCU provides a user interface and control over the dual PLL system composed of PLL0 and PLL1 blocks (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1 as shown in *Chapter 6: Reset and boot*. The key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by PLL1 PHI frequency modulated output.

Note: For chip-specific implementation details, refer to Chapter 6: Reset and boot.

12.2 Block diagram

Figure 55 shows the block diagram of the PLL Digital Interface.

Figure 55. Dual PLL digital interface block diagram



12.3 Features

The Dual PLLDIG has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for reducing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.
- Programmable frequency modulation on PLL1.
- The frequency range after the prescaler is required to be 8-20 Mhz.
- Lock detect circuitry reports when the PLLs have achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
  - User-selectable ability to generate a fault request upon loss of lock.
  - The loss of lock indication is sent to the FCCU.

Note: Refer to the device datasheet for information on the input and output clock frequency range.

12.4 Modes of operation

The operating mode of the PLLs is determined by the value of PLL*n*CR[CLKCFG]. The mode of operation for the PLL is defined below:

Normal mode with reference, and either PLL0 or both PLLs enabled.

Normal mode is defined as the mode where the clocks are driven by the PLL. When using a crystal for the clock reference, reset must remain asserted until the oscillator has stabilized.

When PLL0 is powered down and working in bypass mode, PLL1 must be bypassed and powered down. PLL1 must be configured to work in normal mode only after PLL0 has achieved lock (for example, bypass with PLL0 running or PLL0 in normal mode, PLL*n*SR[LOCK] = 1).

12.4.1 Normal mode with reference, PLL0 or both PLLs enabled

In the normal mode, PLL0 receives an input clock from the reference and the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer recommended external support circuitry and short signal route from the MCU to the crystal.

PLL0 generates a non-modulated clock which drives PLL0_PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, output divider (Reduced Frequency Divider) and whether the PLL1 is modulated or not can be programmed by writing to the PLL1FM register, *Section 12.5.2.7: PLL1 frequency modulation register (PLL1FM)*.

Note: While powering down the PLLs, user must ensure that PLL1 is powered down first followed by powering down PLL0 for proper shut off.

Note: Only when the PLL0 achieves lock must PLL1 be configured to work in either mode by programming the appropriate RCC_CR and RCC_PLLCFCR registers (see Section 12.7: Initialization information).

12.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs.

12.5.1 Memory map

Table 120 shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

Table 120. Dual PLL digital interface memory map

Offset (hex)	Register	Section
0x0000	PLL0 Control Register (PLL0CR)	*Section 12.5.2.1*
0x0004	PLL0 Status Register (PLL0SR)	*Section 12.5.2.2*
0x0008	PLL0 Divider Register (PLL0DV)	*Section 12.5.2.3*
0x000C–0x001F	Reserved	
0x0020	PLL1 Control Register (PLL1CR)	*Section 12.5.2.4*
0x0024	PLL1 Status Register (PLL1SR)	*Section 12.5.2.5*
0x0028	PLL1 Divider Register (PLL1DV)	*Section 12.5.2.6*
0x002C	PLL1 Frequency Modulation Register (PLL1FM)	*Section 12.5.2.7*
0x0030	PLL1 Fractional Divide Register (PLL1FD)	*Section 12.5.2.8*
0x0034–0x003F	Reserved	

12.5.2 Register descriptions

Some of the register reset values are specifically configured for each unique device by external configuration signals or parameters.

12.5.2.1 PLL0 control register (PLL0CR)

Offset: 0x0000 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CLKCFG		0[1]	0[1]	0[1]	0[1]	LOLIE[2]	0[1]	0[1]	0[1]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be written at any time, but writes are ignored. Reads returns previously written value.
2. Refer to the Clocking chapter for details on loss of lock management.

Figure 56. PLL0 control register (PLL0CR)

Table 121. PLL0CR field descriptions

Field	Description
9:8 CLKCFG	Clock configuration This field indicates the operational state of PLL. 00 PLL off. 01 Reserved 10 Reserved. 11 Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits read 11b. **Note:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.
3 LOLIE	Loss of lock FAULT enable The LOLIE bit enables a loss of lock FAULT request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the FAULT request is ignored. The FAULT request only occurs in normal mode. 0 Ignore loss of lock. FAULT not requested 1 Enable FAULT request upon loss of lock

12.5.2.2 PLL0 status register (PLL0SR)

Offset: 0x0004 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	LOLF	LOCK	0	0
W													w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Figure 57. PLL0 status register (PLL0SR)

Table 122. PLL0SR field descriptions

Field	Description
3 LOLF	Loss of lock flag. W1C to erase the FAULT source. This bit provides the FAULT flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit remains set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. FAULT service not requested 1 Loss of lock detected. FAULT service requested
2 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked 1 PLL is locked

12.5.2.3 PLL0 divider register (PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency dividers, pre-divider, and loop divider.
The values of PREDIV and MFD must not be changed when the PLL is on and locked (working in Normal mode). If these fields are changed without powering down the PLL, the PLL loses lock and generates either a reset or an interrupt, based on which is enabled.
The reduced frequency divider fields (RFDPHI, RFDPHI1) can be modified at anytime, but the changes only become effective after PLL0 is disabled, then re-enabled.

Offset: 0x0008 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	RFDPHI1				0	0	0	0	0	RFDPHI					
W																
Reset	0	_(1)	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	PREDIV			0	0	0	0	0	MFD						
W																
Reset	0	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. Refer to the Clocking chapter for default reset value information.

Figure 58. PLL0 divider register (PLL0DV)

Table 123. PLL0DV field descriptions

Field	Description
30:27 RFDPHI1	PHI1 reduced frequency divider This 4-bit field determines the VCO/2 clock post divider for driving the PHI1 output clock. 00xx Reserved 0100 Divide by 4 0101 Divide by 5 ... 1110 Divide by 14 1111 Divide by 15
21:16 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 00 Reserved 01 Divide by 1 02 Divide by 2 03 Divide by 3 ... 3E Divide by 62 3F Divide by 63

Table 123. PLL0DV field descriptions (continued)

Field	Description
14:12 PREDIV	Input clock pre-divider This 3-bit field controls the value of the divider on the input clock. The output of the pre-divider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 ... 110 Divide by 6 111 Divide by 7
6:0 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 00 Reserved ... 07 Reserved 08 Divide by 8 ... 7E Divide by 126 7F Divide by 127

12.5.2.4 PLL1 control register (PLL1CR)

Offset: 0x0020 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0(1)	CLKCFG		0(1)	0(1)	0(1)	0(1)	LOLIE(2)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be written at any time, but writes are ignored. Reads returns previously written value.
2. Refer to the Clocking chapter for details on loss of lock management.

Figure 59. PLL1 control register (PLL1CR)

Table 124. PLL1CR field descriptions

Field	Description
9:8 CLKCFG	Clock configuration This field indicates the operational state of PLL. 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits read 11b. **Note:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.
3 LOLIE	Loss of lock FAULT enable The LOLIE bit enables a loss of lock FAULT request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the FAULT request is ignored. The FAULT request only occurs in normal mode. 0 Ignore loss of lock. FAULT not requested 1 Enable FAULT request upon loss of lock and enable the FCCU failure "Loss of clock flag PLL1"

12.5.2.5 PLL1 status register (PLL1SR)

Offset: 0x0024 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0(1)	0	0	0	LOLF	LOCK	0	0
W											*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Figure 60. PLL1 status register (PLL1SR)

Table 125. PLL1SR field descriptions

Field	Description
3 LOLF	Loss of lock flag. W1C for erasing the FAULT source. This bit provides the FAULT flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit remains set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. FAULT service not requested 1 Loss of lock detected. FAULT service requested
2 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked 1 PLL is locked

12.5.2.6 PLL1 divider register (PLL1DV)

The PLL1DV register provides the output clock reduced frequency dividers and loop divider values.
The value of MFD must not be changed when the PLL is on and locked (working in Normal mode). If the field is changed without powering down the PLL, the PLL loses lock and generates either a reset or an interrupt, based on which is enabled.
The reduced frequency divider field (RFDPHI) can be modified at anytime, but the changes only become effective after PLL1 is disabled, then reenabled.

Offset: 0x0028 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	RFDPHI[1]					
W																
Reset	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	MFD[1]						
W																
Reset	0	0	0	0	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. Refer to the Clocking chapter for default reset value information.

Figure 61. PLL1 divider register (PLL1DV)

Table 126. PLL1DV field descriptions

Field	Description
21:16 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 0x00 Reserved 0x01 Divide by 1 0x02 Divide by 2 ... 0x3E Divide by 62 0x3F Divide by 63
6:0 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 0x10 Divide by 16 0x11 Divide by 17 .... 0x21 Divide by 33 0x22 Divide by 34 All other settings are reserved

12.5.2.7 PLL1 frequency modulation register (PLL1FM)

The PLL1FM register enables frequency modulation on PLL1 and provides controls for the FM spread spectrum, modulation depth and modulation rate. This register must be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down).
Changing the values of PLL1FM[MODEN] and PLL1FM[MODSEL] fields once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset if proper FCCU reaction is programmed. The PLL would then require power cycling to regain normal functionality.
The modulation period (PLL1FM[MODPRD]) and increment step fields (PLL1FM[INCSTP]) can be changed without losing lock, however, the output clock from PLL may have an incorrect frequency for a few cycles after either of these updates (as defined in the datasheet).

Offset: 0x002C Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MODEN	MODSEL	MODPRD												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	INCSTP														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 62. PLL1 frequency modulation register (PLL1FM)

Table 127. PLL1FM field descriptions

Field	Description
30 MODEN	Modulation enable This bit enables the frequency modulation. 0 Frequency modulation disabled 1 Frequency modulation enabled
29 MODSEL	Modulation selection This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0 Modulation centered around nominal frequency 1 Modulation spread below nominal frequency
28:16 MODPRD	Modulation period This 13-bit field is the binary equivalent of the modulation period variable derived from the formula: $MODPRD = round(\frac{f_{ref}}{4 \times f_{mod}})$ where f_{ref} represents the frequency of the feedback divider and f_{mod} represents the modulation frequency (refer to equation in *Section 12.6.3*).
14:0 INCSTP	Increment step This 15-bit field is the binary equivalent of the INCSTP variable derived from the formula: $INCSTEP = round(\frac{(2^{15} - 1) \times MD \times PLL1CR[MFD]}{100 \times 5 \times MODPRD})$ where *md* represents the peak modulation depth in percentage (+/- *MD* for centered modulation, -2**MD* for modulation below nominal frequency), and MFD represents the nominal value of the feedback loop divider (refer to equation in *Section 12.6.3*).

12.5.2.8 PLL1 fractional divider register (PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register must be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of FDEN once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset if proper FCCU reaction is programmed. The PLL would then require power cycling to regain normal functionality.
The dither disable and fractional divider fields can be changed without losing lock, however, the output clock from PLL can have an incorrect frequency for a few cycles after either of these updates.

Offset: 0x0030 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	FDEN	0	0	0	0	0	0	0	0	0	0	0	0	DTHDIS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	FRCDIV											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 63. PLL1 fractional divider register (PLL1FD)

Table 128. PLL1FD field descriptions

Field	Description
30 FDEN	Fractional divide enable This bit enables the fractional divider in the loop divider for PLL1. 0 Fractional divide disabled 1 Fractional divide enabled
17:16 DTHDIS	Dither disable Bit 17 and bit 16 control noise of SSCG and Fractional controller. 00 Increase PLL multiplication factor by 1/2^13 and has maximum noise control of SSCG and Fractional controller. 01 Increase PLL multiplication factor by 1/2^13 and has some noise control of SSCG and Fractional controller. 10 No influence on PLL multiplication factor and has some noise control of SSCG and Fractional controller. 11 No influence on PLL multiplication factor and no noise control of SSCG and Fractional controller.
11:0 FRCDIV	Fractional divide input When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled, the mean VCO clock frequency is given by: $\text{VCO Frequency} = \text{InputClock} \times (\text{PLL1DV[MFD]} + (\text{FRCDIV} \div 2^{12}))$ where FRCDIV is the decimal equivalent of the FRCDIV bits (refer to *Section 12.6.2*). **Note:** Depending on DTHDIS value the multiplication factor can change as described in DTHDIS field description.

12.5.3 Register classification for safety requirements

This module is classified as a safety Vital Module (ViMo). Hence, the registers of this module have been classified as shown in the SR5E1x Reference Manual “Functional Safety” chapter.

12.6 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

12.6.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for each PLL are discussed in detail in the *SR5E1x datasheet*.

12.6.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV and PLL1FD registers, and calculated according to the following equations:

Equation 6

$$f_{pll0_phi} = f_{pll0_ref} \times \frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI]}$$

Equation 7

$$f_{pll0_phi1} = f_{pll0_ref} \times \frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]}$$

Equation 8

$$f_{pll1_phi} = f_{pll1_ref} \times \left(\frac{PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}}}{2 \times PLL1DV[RFDPHI]} \right)$$

Note: Depending on the DTHDIS value the multiplication factor of Equation 8 can change as described in the PLL1DV[DTHDIS] field description (refer to Table 128).

The relationship between the VCO frequency f_{VCO} and the output frequency of the PLLs is determined by the configuration of the PLL0DV, PLL1DV and PLL1FD registers, according to the following equations:

Equation 9

$$f_{pll0_VCO} = \frac{f_{pll0_ref} \times PLL0DV[MFD] \times 2}{PLL0DV[PREDIV]}$$

Equation 10

$$f_{pll1_VCO} = f_{pll1_ref} \times \left(PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} \right)$$

Note: Depending on DTHDIS value the multiplication factor of Equation 10 can change as described in PLL1DV[DTHDIS] field description (refer to Table 128).

Note: f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1.

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification for PLL0 or PLL1 (refer to the device datasheet). Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLL*n*SR[LOCK] flag are immediately negated if the fields of PLL*n*DV are changed without powering down the analog PLLs.

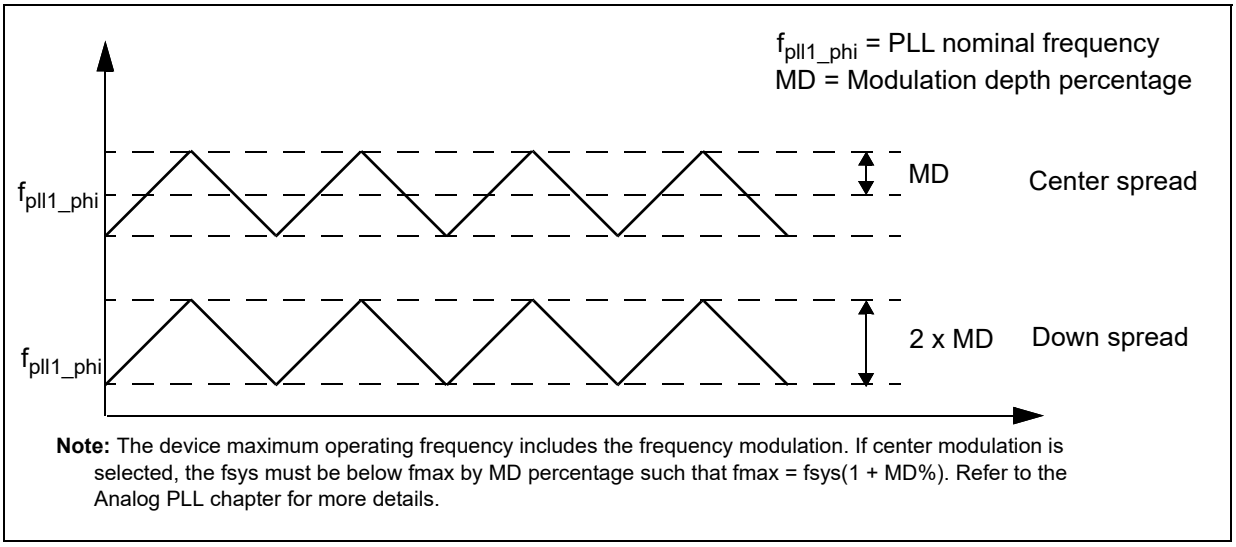
When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLL*n*SR[LOCK] flag is held negated. After the timer expires, the PLL*n*SR[LOCK] flag reflects the value coming from the PLL lock detection circuitry.

The recommended procedure to program the PLLs and engage normal mode is shown in Section 12.7.

12.6.3 Frequency modulation

Frequency modulation uses a triangular profile as shown in Figure 64. The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

Figure 64. Triangular frequency modulation



Note: The device maximum operating frequency includes the frequency modulation. If center modulation is selected, the fsys must be below fmax by MD percentage such that fmax = fsys(1 + MD%). Refer to the Analog PLL chapter for more details.

The following equations define how to calculate PLL1FM[MODPRD] (Equation 11) and PLL1FM[INCSTP] (Equation 12) based on the frequency of the feedback divider (f_{pll1_ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD):

Equation 11

$$PLL1FM[MODPRD] = round\left(\frac{f_{pll1_ref}}{4 \times f_{mod}}\right)$$

Equation 12

$$PLL1FM[INCSTP] = round\left(\frac{(2^{15} - 1) \times MD \times PLL1DV[MFD]}{100 \times 5 \times PLL1FM[MODPRD]}\right)$$

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

Equation 13

$$(PLL1FM[MODPRD] \times PLL1FM[INCSTP]) < 2^{15}$$

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is given by the following formula:

Equation 14

$$ModulationDepth = round\left(\frac{PLL1FM[MODPRD] \times PLL1FM[INCSTP] \times 100 \times 5}{(2^{15} - 1) \times PLL1DV[MFD]}\right)$$

As an example, suppose the following configuration:

- Input frequency: 40 MHz
- Loop divider (PLL1DV[MFD]): 20
- Input divider: 1
- Output divider RFD: 5
- VCO frequency: 40 MHz × 20 = 800 MHz
- PLL output frequency: 800 MHz / 2*RFD = 800 MHz / 10 = 80 MHz
- Center spread (PLL1FM[MODSEL] = 0)
- Modulation frequency: 200 kHz
- Modulation depth: ± 1.9% (3.8% peak-to-peak)
- PLL1FM[MODPRD] = Round[(40 × 10^6) / (4 × 200 × 10^3)] = Round[50] = 50
- PLL1FM[INCSTP] = Round[((2^{15} – 1) × 1.9× 20) / (100 × 5 × 50)] = Round[49.8] = 50
- PLL1FM[MODPRD] × PLL1FM[INCSTP] = 50 × 50 = 2500 (which is less than 2^{15})
- MD (quantized) = ((50 × 50 × 100 × 5) / ((2^{15} – 1) × 20) = 1.90740%

In this example, the modulation error is 0.00740%.

The FM parameters do not get reset when PLL0 loses its lock. These parameters get reset on destructive reset. Whenever PLL0 gets relocked FM modulation starts with the previous values. The sequence for reprogramming the FM is:

1. Power down PLL1[a]
2. Program the PLL1FM while PLL1 is powered down
3. Power up the PLL1[b]

12.7 Initialization information

Coming out of reset the PLLs are disabled. The recommended procedure to program the PLLs is:

1. Configure PLL0 and related modules.
   a) With PLL0 disabled, program PLL0 clock source in RCC_PLLCFGR[PLL0SRC]. Default source is none.
   b) Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].
   c) If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. XOSC frequency is compared to the IRCOSC source when XOSC is turned on.
2. Turn on XOSC and PLL0.
   a) Set bit RCC_CR[HSEON] and wait for RCC_CR[HSERDY].
   b) Set bit RCC_CR[PLL0ON] and wait for RCC_CR[PLL0LOCK].
3. Select system clock if you want PLL0 as system clock.
   a) Out of reset and up to this point the system clock is 16 MHz HSI. If desired, it is possible to switch system clock to PLL0, by writing RCC_CFGR[SW] and wait until RCC_CFGR[SWS] == RCC_CFGR[SW]. This is not the recommended clock configuration.
4. Configure PLL1 and related modules
   a) With PLL1 disabled, program PLL1 clock source in RCC_PLLCFGR[PLL1SRC]. Default is none.
   b) Program PLL1DV[MFD] and PLL1DV[RFDPHI].
   c) If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.
5. Turn on PLL1
   a) Set bit RCC_CR[PLL1ON] and wait for RCC_CR[PLL1LOCK].
6. Select PLL1 as system clock.
   a) Out of reset and up to this point the system clock is 16 MHz HSI.
   b) Write RCC_CFGR[SW] and wait until RCC_CFGR[SWS] == RCC_CFGR[SW]. This is the recommended clock configuration.

a. By writing RCC_CR [PLL1ON] = 0b.

b. By writing RCC_CR [PLL1ON] = 1b.

13 Clock monitor unit (CMU)

13.1 Introduction

The Clock monitor unit (CMU), also referred to as clock quality checker or clock fault detector, serves three purposes:

- Measures the frequency of clock sources CLKMT0_RMN/CLKMT1 with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and also detects if the monitored clock frequency leaves an upper or lower frequency boundary

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which can take the necessary corrective actions as its configuration dictates.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the division factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate events when the CLKMN1 frequency is less than the CLKMT0_RMN frequency divided by four or if CLKMN1 leaves an upper or lower frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of one clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN, CLKMT1), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

Note: Refer to the clocking chapter for the number of instances of the CMU in this device.

13.1.1 Main features

- CLKMT0_RMN, CLKMT1 frequency measurement with CLKMN0_RMT as the reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN/n clock.
- Upper or lower frequency boundary monitoring of CLKMN1 with respect to CLKMT0_RMN / 4.
- Event generation for various failures detected inside monitoring unit.

13.2 Block diagram

The block diagram of the CMU module(s) is shown in the clocking chapter of this reference manual.

13.3 Signals

Table 129 describes the signals on the boundary of the CMU (in alphabetical order).

Table 129. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference: Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1: Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference: Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.
CLKMT1	I	Metered Clock Signal 1: Receives a clock signal that the CMU measures against a reference clock frequency.

Note: Refer to the clocking chapter for device specific clock sources of each CMU.

13.4 Memory map and register definition

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in *Table 130*.

Table 130. CMU memory map

Address offset	Register	Location
0x0000	CMU Control Status Register (CMU_CSR)	*Section 13.4.1.1*
0x0004	CMU Frequency Display Register (CMU_FDR)	*Section 13.4.1.2*
0x0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	*Section 13.4.1.3*
0x000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	*Section 13.4.1.4*
0x0010	CMU Interrupt Status Register (CMU_ISR)	*Section 13.4.1.5*
0x0014–0x0017	Reserved	
0x0018	CMU Measurement Duration Register (CMU_MDR)	*Section 13.4.1.6*

Note: See Clocking chapter for register and field availability details.

13.4.1 Register descriptions

13.4.1.1 CMU Control Status Register (CMU_CSR)

Offset: 0x0000 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	SFM(1)	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CKSEL1(1)		0	0	0	0	0	RCDIV(1)		CME
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Not all CMU blocks utilize this feature. Refer to the clocking chapter for device specific CMU implementation details.

Figure 65. CMU Control Status Register (CMU_CSR)

Table 131. CMU_CSR field descriptions

Field	Description
23 SFM	Start Frequency Measure The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR register. CMU_MDR[MD] must be written before enabling frequency measurement (CMU_CSR[SFM] = 1). Do not write CMU_MDR[MD] while CMU_CSR[SFM] = 1. Software must only read the value of CMU_FDR[FD] bits once the hardware has cleared the CMU_CSR[SFM] bit. 0 Frequency measurement is completed or not yet started. 1 Frequency measurement is not completed.
9:8 CKSEL1	Frequency measure clock selection bit CKSEL1 selects the clock to be measured by the frequency meter. This only effects CMU instances that utilizes clock metering. Not all CMU blocks utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected. 01 CLKMT1 is selected. 10 Reserved. 11 CLKMT0_RMN is selected.

Table 131. CMU_CSR field descriptions (continued)

Field	Description
2:1 RCDIV	CLKMT0_RMN division factor These bits specify the CLKMT0_RMN division factor. The output clock frequency is f_{CLKMT0_RMN} divided by the factor 2^{RCDIV}. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. Ensure clock source for CLKMN0_RMT is off before writing CMU_CSR[RCDIV] otherwise the operation can lead to a spurious OLR event. The clock division coding is as follows: 00 CLKMT0_RMN divided by 1 (No division). 01 CLKMT0_RMN divided by 2. 10 CLKMT0_RMN divided by 4. 11 CLKMT0_RMN divided by 8.
0 CME	CLKMN1 monitor enable CMU_HFREF and CMU_LFREF registers must be written before enabling the clock monitoring (CMU_CSR[CME] =1) 0 CLKMN1 monitor is disabled. 1 CLKMN1 monitor is enabled.

13.4.1.2 CMU Frequency Display Register (CMU_FDR)

The FDR is used to determine the measured frequency being monitored by the CMU.

Offset: 0x0004 Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	FD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 66. CMU Frequency Display Register (CMU_FDR)

Table 132. CMU_FDR field descriptions

Field	Description
19:0 FD	Measured frequency bits This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMN0_RMT}). The measured value is given by the following formula: $f_{sel} = (f_{CLKMN0_RMT} \times CMU_MDR[MD]) / CMU_FDR[FD]$

13.4.1.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU uses for comparing against the monitored clock. *Figure 67* and *Table 133* show the CMU_HFREFR register.

Offset: 0x0008 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	HFREF											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 67. CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

Table 133. CMU_HFREFR field descriptions

Field	Description
11:0 HFREF	High Frequency reference value These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: (HFREF / 16) × (f_{CLKMT0_RMN} / 4)

13.4.1.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU uses for comparing against the monitored clock. *Figure 68* and *Table 134* show the CMU_LFREFR register.

Offset 0x000C Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	LFREF											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 68. CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

Table 134. CMU_LFREFR field descriptions

Field	Description
11:0 LFREF	Low Frequency reference value These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: (LFREF / 16) × (f_{CLKMT0_RMN} / 4)

13.4.1.5 CMU Interrupt Status Register (CMU_ISR)

Offset: 0x0010 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	FLCI	FHHI	FLLI	OLRI(1)
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Not all CMU blocks utilize this feature. See Clocking chapter for device specific CMU implementation details.

Figure 69. CMU Interrupt Status Register (CMU_ISR)

Table 135. CMU_ISR field descriptions

Field	Description
3 FLCI	CLKMN1 frequency less than reference clock event status This bit is set by hardware when CLKMN1 frequency becomes lower than reference clock frequency (f_{CLKMT0_RMN}/4) value and CLKMN1 is 'ON' as signaled by the RCC. It can be cleared by software by writing '1'. 0 No FLC event 1 FLC event occurred
2 FHHI	CLKMN1 frequency higher than high reference event status This bit is set by hardware when CLKMN1 frequency becomes higher than HFREF value and CLKMN1 is 'ON' as signaled by the RCC. It can be cleared by software by writing '1'. 0 No FHH event 1 FHH event occurred
1 FLLI	CLKMN1 frequency less than low reference event status This bit is set by hardware when CLKMN1 frequency becomes lower than LFREF value and CLKMN1 is 'ON' as signaled by the RCC. It can be cleared by software by writing '1'. 0 No FLL event 1 FLL event occurred
0 OLRI	Oscillator frequency less than f_{CLKMT0_RMN} / 2^{RCDIV} event status This bit is set by hardware when the f_{CLKMN0_RMT} is less than f_{CLKMT0_RMN} / 2^{RCDIV} frequency and CLKMN0_RMT is 'ON' as signaled by the RCC. It can be cleared by software by writing '1'. 0 No OLR event 1 OLR event occurred

Note: All the flags in CMU_ISR are set asynchronously. This register must be read only after a fault event is triggered by the CMU and mapped on FCCU.

Otherwise, the read access on this register may fetch an incorrect value.

Before entering low-power stop modes, all clock frequency measurements in CMU must be disabled. Failing to do so may result in spurious interrupts.

13.4.1.6 CMU Measurement Duration Register (CMU_MDR)

Offset: 0x0018 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	MD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 70. CMU Measurement Duration Register (CMU_MDR)

Table 136. CMU_MDR field descriptions

Field	Description
19:0 MD	Measurement duration bits This field displays the measurement duration in terms of selected clock (CLKMT0_RMN, CLKMT1) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1. The application software must program CMU_MDR[MD] with a value that does not saturate CMU_FDR[FD] to 20'hFFFFF.

13.5 Functional description

This section describes the functionality of the CMU.

13.5.1 Frequency meter

The purpose of frequency meter is to evaluate the deviation from the nominal metered source (such as, CLKMT0_RMN, CLKMT1) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the Clocking chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR as shown in the following equation:

Equation 15 $f_{sel} = (f_{CLKMN0_RMT} \times MDR[MD]) / FDR[FD]$

13.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than frequency of CLKMT0_RMN divided by $2^{CMU_CSR[RCDIV]}$ and CLKMN0_RMT is 'ON' as signaled by the RCC, then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

13.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If fCLKMN1 is greater than the reference value determined by field CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the RCC, then:

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

Note: The high reference value must be programmed considering the following factors:

- *2 cycles of f_{CLKMN1} which is the built-in tolerance of the monitor implementation.*
- *Frequency variation% of f_{CLKMT0_RMN} across PVT.*

Note: An example of determining the HFREFActual value is as follows. Assume a fCLKMT0_RMN = 16 MHz with an accuracy of +/-5%. In order to monitor fCLKMN1 = 200 MHz, the ideal HFREFIdeal = 800. The actual HFREF value is 842 when the accuracy is taken into consideration (HFREFActual = (HFREFIdeal + 2) × 1.05).

If f_{CLKMN1} is less than a reference clock frequency (f_{CLKMT0_RMN}/4) and the CLKMN1 is 'ON' as signaled by the RCC, then:

The CMU writes 1 to kCMU_ISR[FLCI].

f_{CLKMN1} must be greater than f_{CLKMT0_RMN} / 4 by at least 0.5 MHz in order to guarantee the correct CLKMN1 monitoring.

If fCLKMN1 is lower than CMU_LFREFR[LFREF] and CLKMN1 is 'ON' as signaled by the RCC, then:

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note: The low reference value must be programmed considering the following factors:

- *Two cycles of f_{CLKMN1} which is the built-in tolerance of the monitor implementation*
- *Frequency variation % of f_{CLKMT0_RMN} across PVT*

For example, assuming f_{CLKMT0_RMN} is 16 MHz with +/-5% variation, in order to monitor the f_{CLKMN1} (200 MHz), the ideal $LFREF_{Ideal}$ is 800. The adjusted value considering the tolerances is 758 ($LFREF_{Actual}$ = ($LFREF_{Ideal}$ - 2) x 0.95).

The minimum value that can be programmed in CMU_LFREFR[LFREF] is 3. The minimum frequency that can be monitored using the CLKMN1 supervisor is >= f_{CLKMT0_RMN} / 16.

14 Power management controller digital interface (PMC_Dig)

14.1 Introduction

The Control PMU manages the PMU at reset and during run. It checks the LVD or HVD accuracy.

The PMC analog block interfaces the digital part of the device by the PMC_Dig block.

This chapter describes the digital block of the PMU.

The PMC_Dig contains all of the registers and digital logic to generate all of the enable signals, two stage trim control bits, power on reset (POR) generation and test mode logic for the analog block. The PMU digital logic also contains the controls for PMU analog outputs to be sensed with the ADC module.

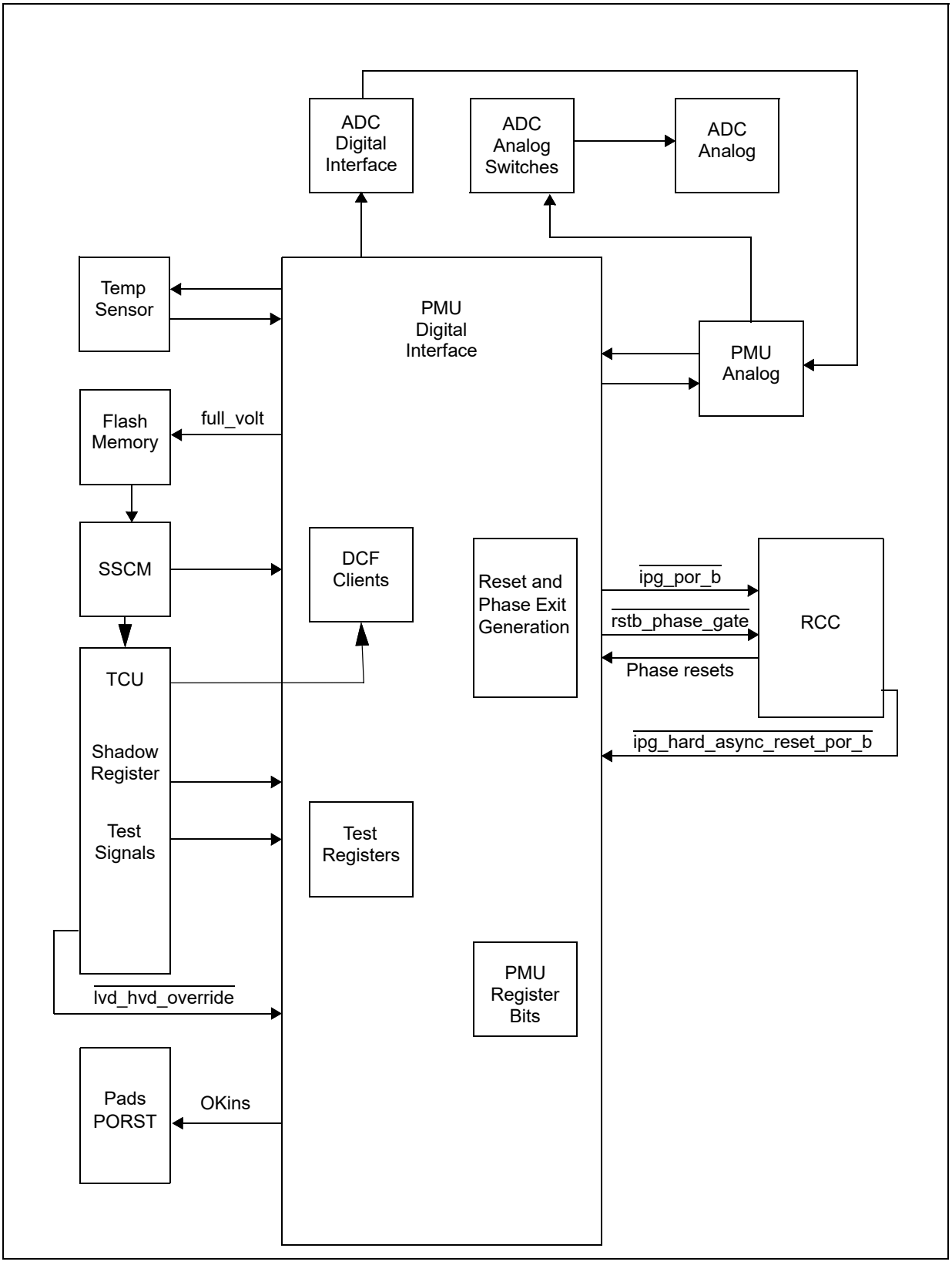
In addition, the PMU digital logic contains the enable signals, trim bits and other registers for the temperature sensor for SOC and a dedicated temperature sensor for RC regulator.

There are also several custom interfaces to various other blocks: the Reset and Clock Control (RCC) block receives the LVD values during POR to use for phase transition conditions.

There is one custom interface to Flash memory via the SSCM (and DCF client blocks) that is used to load the trim values during initial POR.

The PMU digital block generates a transfer error event to the system if the selected address is out of range of PMU digital memory map. A transfer error is not generated if an unused address within the PMU digital memory space is accessed. *Figure 71* shows the top level block diagram of the digital PMU and its interfaces to the device modules.

Figure 71. Digital PMU block diagram



14.2 Main features

- PMU management at a main reset
- Resets signal generation

14.2.1 Standard features

- IPS bus interface

14.3 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- Byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data write/read operations to any registers
- Any other operation (free byte enables or other operations) must be avoided

The PMC_Dig block generates a transfer error in the following cases:

- The PMU digital block generates a transfer error event to the system if the selected address is out of range of PMU digital memory map. A transfer error is not generated if an unused address within the PMU digital memory space is accessed
- Any write/read operation different from byte/hword/word (free byte enables or other operations) on each register

The registers of the PMC_Dig block are accessible (read/write) in each access mode: user, supervisor or test.

About the reset used among the registers:

- The POR reset is used on EPR_LVx, EPR_HVx, REE_LVx, REE_HVx, RES_LVx and RES_HVx registers
- The destructive reset is used on REE_TD, RES_TD, FEE_LVx, FEE_HVx and FEE_TD registers
- The functional reset is used on IE_HVx, IE_LVx, IE_G, CTL_TD registers

14.4 Memory map/register definition

Table 137 shows the PMU memory map. The PMU includes many registers for configuring and monitoring LVD monitors and some other control registers.

Table 137. Power Management Controller Memory Map

Offset	Register name	Location
0x0000	Event Pending Register LV0 (EPR_LV0)	*Section 14.5.1*
0x0004	Reset Event Enable LV0 (REE_LV0)	*Section 14.5.2*
0x0008	Reset Event Select LV0 (RES_LV0)	*Section 14.5.3*

Table 137. Power Management Controller Memory Map (continued)

Offset	Register name	Location
0x000C	Interrupt Enable LV0 (IE_LV0)	*Section 14.5.4*
0x0010	FCCU Event Enable LV0 (FEE_LV0)	*Section 14.5.5*
0x0014–0x001F	***Reserved***	
0x0020	Event Pending Register LV1 (EPR_LV1)	*Section 14.5.6*
0x0024	Reset Event Enable LV1 (REE_LV1)	*Section 14.5.7*
0x0028	Reset Event Select LV1 (RES_LV1)	*Section 14.5.8*
0x002C	Interrupt Enable LV1 (IE_LV1)	*Section 14.5.9*
0x0030	FCCU Event Enable LV1 (FEE_LV1)	*Section 14.5.10*
0x0034–0x003F	***Reserved***	
0x0040	Event Pending Register HV0 (EPR_HV0)	*Section 14.5.11*
0x0044	Reset Event Enable HV0 (REE_HV0)	*Section 14.5.12*
0x0048	Reset Event Selection HV0 (RES_HV0)	*Section 14.5.13*
0x004C	Interrupt Enable HV0 (IE_HV0)	*Section 14.5.14*
0x0050	FCCU Event Enable HV0 (FEE_HV0)	*Section 14.5.15*
0x0054–0x007F	***Reserved***	
0x0080	Supply Gauge Status Register (GR_S)	*Section 14.5.16*
0x0084	Pending Gauge Status Register (GR_P)	*Section 14.5.17*
0x0088	Global Interrupt Enable Register (IE_G)	*Section 14.5.18*
0x008C–0x0243	***Reserved***	
0x0244	HPREG SMPS Sel Status Register (HPREG_SMPS_SEL_S)	*Section 14.5.19*
0x0248–0x02FF	***Reserved***	
0x0300	Temperature Event Pending register (EPR_TD)	*Section 14.5.20*
0x0304	Temperature Reset Event Enable register (REE_TD)	*Section 14.5.21*
0x0308	Temperature Reset Event Selection register (RES_TD)	*Section 14.5.22*
0x030C	Temperature sensor configuration register (CTL_TD)	*Section 14.5.23*
0x0310–0x0317	***Reserved***	
0x0318	Temperature FCCU Event Enable register (FEE_TD)	*Section 14.5.24*
0x031C–0x03C7	***Reserved***	
0x03C8	BIST Mask Status register (BIST_MASK_STATUS)	*Section 14.5.25*
0x03CC	User BIST Flags register phase1(BIST_FLAGS_PHASE1)	*Section 14.5.26*

Table 137. Power Management Controller Memory Map (continued)

Offset	Register name	Location
0x03D0	User BIST Flags register phase2(BIST_FLAGS_PHASE2)	*Section 14.5.27*
0x03D4	User BIST Control register (BIST_CTRL)	*Section 14.5.28*
0x03D8	User BIST Time1 and Time0 register (BIST_TIME10)	*Section 14.5.29*
0x03DC	User BIST Time3 and Time2 register (BIST_TIME32)	*Section 14.5.30*
0x03E0	User BIST Time6 and Time5 register (BIST_TIME65)	*Section 14.5.31*
0x03E4	User BIST VD Under Test Monitor Register (BIST_DEBUG)	*Section 14.5.32*

Table 138 shows how the 32-bit registers for LV0, LV1, and HV0 are organized.

Table 138. 32-bit registers LV0, LV1, HV0 arrangement table

Register class	BYTE 3[1] (bits [31:24])	BYTE 2[1] (bits [23:16])	BYTE 1[1] (bits [15:8])	BYTE 0[1] (bits [7:0])
LV0	VD3 = LVD119_C_FL_DD LVD119_RC_PLL0/1	VD2 = Reserved	VD1 = Reserved	VD0 = Reserved
LV1	VD7 = UVD145_RC	VD6 = HVD140_C	VD5 = Not implemented	VD4 = Not implemented
HV0	VD11 = LVD290_AD_AS LVD290_FL_C_IO0_IO1	VD10 = LVD290_DACCMP_OSC UVD380_AS_DACCMP	VD9 = Not implemented	VD8 = Reserved

1. Refer to *Section 14.5: Registers description*.

Table 139 shows the correspondence between VD number and VD name, where HV means a 3.3 V supply and LV means a 1.2 V supply.

Table 139. VD number versus VD name

VD number	VD name	VD meaning
VD0	POR031_C	LV detector asserts when PMU supply < 0.31 V
VD1	MVD102T_C	LV detector asserts when PMU supply < 1.02 V
VD2	MVD114_C	LV detector asserts when PMU supply < 1.14 V
VD2	MVD114_FA	LV detector asserts when Flash supply < 1.14 V
VD3	LVD119_C	LV detector asserts when PMU supply < 1.19 V
VD3	LVD119_FL	LV detector asserts when Flash supply < 1.19 V
VD6	HVD140_C	LV detector asserts when PMU supply > 1.40 V

Table 139. VD number versus VD name (continued)

VD number	VD name	VD meaning
VD7	UVD145_C	LV detector asserts when PMU supply > 1.45 V
VD7	UVD145_RC	LV detector asserts when RCOSC supply > 1.45 V
VD8	POR200_C	HV detector asserts when PMU supply < 2.00 V
VD9	MVD240T_C	HV detector asserts when PMU supply < 2.40 V
VD10	MVD270_C	HV detector asserts when PMU supply < 2.70 V
VD10	MVD270_FL	HV detector asserts when Flash supply < 2.70 V
VD11	LVD290_DACCMP	HV detector asserts when DAC/CMP supply < 2.90V
VD11	LVD290_OSC	HV detector asserts when OSC supply < 2.90V
VD11	LVD290_AD	HV detector asserts when sigma delta ADC supply < 2.90 V
VD11	LVD290_AS	HV detector asserts when SAR ADC supply < 2.90 V
VD11	LVD290_C	HV detector asserts when PMU supply < 2.90 V
VD11	LVD290_FL	HV detector asserts when Flash supply < 2.90 V
VD11	LVD290_IO0	HV detector asserts when IO RING0 supply < 2.90 V
VD11	LVD290_IO1	HV detector asserts when IO RING1 supply < 2.90 V
VD15	UVD380_C	HV detector asserts when PMU supply > 3.80 V
VD15	UVD380_FL	HV detector asserts when Flash supply > 3.80 V
VD15	UVD380_IO0	HV detector asserts when IO RING0 supply > 3.80 V
VD15	UVD380_AS	HV detector asserts when SARADC supply > 3.80 V
VD15	UVD380_DACCMP	HV detector asserts when DAC/CMP supply > 3.80V

14.5 Registers description

14.5.1 Event pending register (EPR_LV0)

This Event Pending Register indicates the present and past state of the voltage detect signals in LOW VOLTAGE 0 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Offset: 0x0000 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LVD3_PLL1	LVD3_PLL0	LVD3_RC	0	LVD3_DD	LVD3_FL	0	LVD3_C	0	0	0	0	0	0	0	0
W	w1c	w1c	w1c		w1c	w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 72. Event Pending Register (EPR_LV0)

Table 140. EPR_LV0 field descriptions

Field	Description
31 LVD3_PLL1	LVD3_PLL1 flag. This bit is the low-voltage status flag associated with LVD119_PLL1 VD, the voltage level detect for the low voltage PLL1 1.19 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.19 V point supply
30 LVD3_PLL0	LVD3_PLL0 flag. This bit is the low-voltage status flag associated with LVD119_PLL0 VD, the voltage level detect for the low voltage PLL0 1.19 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.19 V point supply
29 LVD3_RC	LVD3_RC flag. This bit is the low-voltage status flag associated with LVD119_RC VD, the voltage level detect for the low voltage RCOSC 1.19 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.19 V point supply
27 LVD3_DD	LLVD3_DD flag. This bit is the low-voltage status flag associated with LVD119_DD VD, the voltage level detect for the low voltage DLL 1.19 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.19 V point supply

Table 140. EPR_LV0 field descriptions (continued)

Field	Description
26 LVD3_FL	LVD3_FL flag. This bit is the low-voltage status flag associated with LVD119_FL VD, the voltage level detect for the low voltage Flash 1.19 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.19 V point supply
24 LVD3_C	LVD3_C flag This bit is the low-voltage status flag associated with LVD119_C VD, the voltage level detect for the low voltage CORE 1.19 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.19 V point supply

14.5.2 Reset event enable register (REE_LV0)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0004 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REE3_PLL1	REE3_PLL0	REE3_RC	0	REE3_DD	REE3_FL	0	REE3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 73. Reset Event Enable Register (REE_LV0)

Table 141. REE_LV0 field descriptions

Field	Description
31 REE3_PLL1	REE3_PLL1 reset enable This bit defines whether LVD119_PLL1 VD, the voltage level detect for the low voltage of 1.19 V on PLL1 supply, generates a system reset. The RES3_PLL1 bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
30 REE3_PLL0	REE3_PLL0 reset enable This bit defines whether LVD119_PLL0 VD, the voltage level detect for the low voltage of 1.19 V on PLL0 supply, generates a system reset. The RES3_PLL0 bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
29 REE3_RC	REE3_RC reset enable This bit defines whether LVD119_RC VD, the voltage level detect for the low voltage of 1.19 V on RCOSC supply, generates a system reset. The RES3_RC bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
27 REE3_DD	REE3_DD reset enable This bit defines whether LVD119_DD VD, the voltage level detect for the low voltage of 1.19 V on DLL supply, generates a system reset. The RES3_DLL bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
26 REE3_FL	REE3_FL reset enable This bit defines whether LVD119_FL VD, the voltage level detect for the low voltage of 1.19 V on Flash supply, generates a system reset. The RES3_FL bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
24 REE3_C	REE3_C reset enable This bit defines whether LVD119_C VD, the voltage level detect for the low voltage Core 1.19 V supply, generates a system reset. The RES3_C bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

14.5.3 Reset event select register (RES_LV0)

This Reset Event Select Register controls, in LOW VOLTAGE 0 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled,

the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0008 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RES3_PLL1	RES3_PLL0	RES3_RC	0	RES3_DD	RES3_FL	0	RES3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 74. Reset Event Select Register (RES_LV0)

Table 142. RES_LV0 field descriptions

Field	Description
31 RES3_PLL1	RES3_PLL1 reset event select This bit defines whether LVD119_PLL1 VD, the voltage level detect for the low voltage PLL1 1.19 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_PLL1 bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
30 RES3_PLL0	RES3_PLL0 reset event select This bit defines whether LVD119_PLL0 VD, the voltage level detect for the low voltage PLL0 1.19 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_PLL0 bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
29 RES3_RC	RES3_RC reset event select This bit defines whether LVD119_RC VD, the voltage level detect for the low voltage RCOSC 1.19 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_RC bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

Table 142. RES_LV0 field descriptions (continued)

Field	Description
27 RES3_DD	RES3_DD reset event select This bit defines whether LVD119_DD VD, the voltage level detect for the low voltage DLL 1.19 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_DD bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
26 RES3_FL	RES3_FL reset event select This bit defines whether LVD119_FL VD, the voltage level detect for the low voltage Flash 1.19 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_FL bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
24 RES3_C	RES3_C reset event select This bit defines whether LVD119_C VD, the voltage level detect for the low voltage Core 1.19 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_C bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

14.5.4 Interrupt enable (IE_LV0)

This configuration register contains a set of LV0 interrupt Enable bits that correspond to each of the LV0 Event Pending Registers (EPR_LV0). These bits indicate whether an interrupt is enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-only monitor in the register (IE_G) at address 0x0088. The IE_LV0 bits can be read or written at any time.

Offset: 0x000C Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VD3IE_PLL1	VD3IE_PLL0	VD3IE_RC	0	VD3IE_DD	VD3IE_FL	0	VD3IE_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 75. Interrupt Enable Register (IE_LV0)

Table 143. IE_LV0 field descriptions

Field	Description
31 VD3IE_PLL1	VD3IE_PLL1 Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD119_PLL1 VD, the voltage level detect for the low voltage PLL1 1.19 V supply event. 0 Interrupt disabled 1 Interrupt enabled
30 VD3IE_PLL0	VD3IE_PLL0 Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD119_PLL0 VD, the voltage level detect for the low voltage PLL0 1.19 V supply event. 0 Interrupt disabled 1 Interrupt enabled
29 VD3IE_RC	VD3IE_RC Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD119_RC VD, the voltage level detect for the low voltage RCOSC 1.19 V supply event. 0 Interrupt disabled 1 Interrupt enabled
27 VD3IE_DD	VD3IE_DD Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD119_DD VD, the voltage level detect for the low voltage DLL 1.19 V supply event. 0 Interrupt disabled 1 Interrupt enabled
26 VD3IE_FL	VD3IE_FL Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD119_FL VD, the voltage level detect for the low voltage Flash 1.19 V supply event. 0 Interrupt disabled 1 Interrupt enabled
24 VD3IE_C	VD3IE_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD119_C VD, the voltage level detect for the low voltage Core 1.19 V supply event. 0 Interrupt disabled 1 Interrupt enabled

14.5.5 FCCU event enable register (FEE_LV0)

This FCCU Event Enable Register LV0 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0010 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FEE3_PLL1	FEE3_PLL0	FEE3_RC	0	FEE3_DD	FEE3_FL	0	FEE3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 76. FCCU Event Enable Register (FEE_LV0)

Table 144. FEE_LV0 field descriptions

Field	Description
31 FEE3_PLL1	FEE3_PLL1 FCCU event enable This bit defines whether a VD assertion of LVD119_PLL1 VD, the voltage level detect for the low voltage PLL1 1.19 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the PLL1 does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the PLL1 causes an FCCU event.
30 FEE3_PLL0	FEE3_PLL0 FCCU event enable This bit defines whether a VD assertion of LVD119_PLL0 VD, the voltage level detect for the low voltage PLL0 1.19 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the PLL0 does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the PLL0 causes an FCCU event.
29 FEE3_RC	FEE3_RC FCCU event enable This bit defines whether a VD assertion of LVD119_RC VD, the voltage level detect for the low voltage RCOSC 1.19 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the RCOSC does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the RCOSC causes an FCCU event.
27 FEE3_DD	FEE3_DD FCCU event enable This bit defines whether a VD assertion of LVD119_DD VD, the voltage level detect for the low voltage DLL 1.19 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the DLL does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the DLL causes an FCCU event.

Table 144. FEE_LV0 field descriptions (continued)

Field	Description
26 FEE3_FL	FEE3_FL FCCU event enable This bit defines whether a VD assertion of LVD119_FL VD, the voltage level detect for the low voltage Flash 1.19 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the Flash does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the Flash causes an FCCU event.
24 FEE3_C	FEE3_C FCCU event enable This bit defines whether a VD assertion of LVD119_C VD, the voltage level detect for the low voltage Core 1.19 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the Core does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the Core causes an FCCU event.

14.5.6 Event pending register (EPR_LV1)

This Event Pending Register indicates the present and past state of the voltage detect signals in LOW VOLTAGE 1 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a ‘1’ must be written to the flag bit that needs to be cleared.

Offset: 0x0020 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	UVD7_RC	0	0	0	0	0	0	0	0	0	0	0	0	HVD6_C
W			w1c													w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 77. Event Pending Register (EPR_LV1)

Table 145. EPR_LV1 field descriptions

Field	Description
29 UVD7_RC	UVD7_RC flag This bit is the low-voltage status flag associated with UVD145_RC VD, the voltage level detect for the low voltage RCOSC 1.45 V supply. It is asserted when the supply rises above its corresponding Voltage Detector 7 threshold, and clears when the falls below above its corresponding UVD7 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.45 V point supply
16 HVD6_C	HVD6_C flag This bit is the low-voltage status flag associated with HVD140_C VD, the voltage level detect for the low voltage Core 1.40 V supply. It is asserted when the supply rises above its corresponding Voltage Detector 6 threshold, and clears when the supply falls below its corresponding HVD6 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.40 V point supply

14.5.7 Reset event enable register (REE_LV1)

This Reset Event Enable Register controls, in LOW VOLTAGE 1 range, whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0024 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	REE7_RC	0	0	0	0	0	0	0	0	0	0	0	0	REE6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 78. Reset Event Enable Register (REE_LV1)

Table 146. REE_LV1 field descriptions

Field	Description
29 REE7_RC	REE7_RC reset enable This bit defines whether a UVD145_RC VD, the voltage level detect for the low voltage RCOSC 1.45 V supply of the cold point, generates a system reset. RES_LV1[RES7_RC] determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.
16 REE6_C	REE6_C reset enable This bit defines whether a HVD140_C VD, the voltage level detect for the low voltage Core 1.40 V supply of the cold point, generates a system reset. RES_LV1[RES6_C] determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

14.5.8 Reset event select register (RES_LV1)

This Reset Event Select Register controls, in LOW VOLTAGE 1 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0028 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	RES7_RC	0	0	0	0	0	0	0	0	0	0	0	0	RES6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 79. Reset Event Select Register (RES_LV1)

Table 147. RES_LV1 field descriptions

Field	Description
29 RES7_RC	RES7_RC event select This bit defines whether a UVD145_RC VD, the voltage level detect for the low voltage RCOSC 1.45 V supply of the cold point, generates a destructive reset or a functional reset. REE_LV1[REE7_RC] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
16 RES6_C	RES6_C event select This bit defines whether a HVD140_C VD, the voltage level detect for the low voltage Core 1.40 V supply of the cold point, generates a destructive reset or a functional reset. REE_LV1[REE6_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

14.5.9 Interrupt enable (IE_LV1)

This configuration register contains a set of LV1 interrupt Enable bits that correspond to each of the LV1 Event Pending Registers (EPR_LV1). These bits indicate whether an interrupt is enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-only monitor in the register (IE_G) at address 0x0088. The IE_LV1 bits can be read or written at any time.

Offset: 0x002C Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	UVD7IE_RC	0	0	0	0	0	0	0	0	0	0	0	0	HVD6IE_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 80. Interrupt Enable Register (IE_LV1)

Table 148. IE_LV1 field descriptions

Field	Description
29 UVD7IE_RC	UVD7IE_RC Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the UVD145_RC VD, the voltage level detect for the low voltage RCOSC 1.45 V supply the voltage event occurs. 0 Interrupt disabled 1 Interrupt enabled
16 HVD6IE_C	HVD6IE_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the HVD140_C VD, the voltage level detect for the low voltage Core 1.40 V supply the voltage event occurs. 0 Interrupt disabled 1 Interrupt enabled

14.5.10 FCCU event enable register (FEE_LV1)

This FCCU Event Enable Register LV1 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0030 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	FEE7_RC	0	0	0	0	0	0	0	0	0	0	0	0	FEE6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 81. FCCU Event Enable Register (FEE_LV1)

Table 149. FEE_LV1 field descriptions

Field	Description
29 FEE7_RC	FEE7_RC FCCU event enable This bit defines whether a UVD145_RC VD assertion of RCOSC 1.45 V supply generates an FCCU event. 0 Disabled. A VD assertion of LV1 on the supply of the RCCU does not cause an FCCU event. 1 Enabled. A VD assertion of LV1 on the supply of the RCOSC causes an FCCU event.
16 FEE6_C	FEE6_C FCCU event enable This bit defines whether a HVD140_C VD assertion of Core 1.40 V supply generates an FCCU event. 0 Disabled. A VD assertion of LV1 on the supply of the Core does not cause an FCCU event. 1 Enabled. A VD assertion of LV1 on the supply of the Core causes an FCCU event.

14.5.11 Event pending register (EPR_HV0)

This Event Pending Register indicates the present and past state of the voltage detect signals in HIGH VOLTAGE 0 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a ‘1’ must be written to the flag bit that needs to be cleared.

Offset: 0x0040 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	LVD11_AD	LVD11_AS	LVD11_IO0	LVD11_IO1	0	LVD11_FL	LVD11_C	0	0	0	0	UVD15_DACCMP	UVD15_AS	LVD11_OSC	LVD11_DACCMP
W		w1c	w1c	w1c	w1c		w1c	w1c					w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 82. Event Pending Register (EPR_HV0)

Table 150. EPR_HV0 field descriptions

Field	Description
30 LVD11_AD	LVD11_AD flag This bit is the high-voltage status flag associated with LVD290_AD, the voltage level detect for the high voltage Sigma Delta ADC 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 2.90 V point supply
29 LVD11_AS	LVD11_AS flag This bit is the high-voltage status flag associated with LVD290_AS, the voltage level detect for the high voltage SAR ADC 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 2.90 V point supply
28 LVD11_IO0	LVD11_IO0 flag This bit is the high-voltage status flag associated with LVD290_IO0, the voltage level detect for the high voltage IO Ring0 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 2.90 V point supply
27 LVD11_IO1	LVD11_IO1 flag This bit is the high-voltage status flag associated with LVD290_IO1, the voltage level detect for the high voltage IO Ring1 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 2.90 V point supply
25 LVD11_FL	LVD11_FL flag This bit is the high-voltage status flag associated with LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 2.90 V point supply
24 LVD11_C	LVD11_C flag This bit is the high-voltage status flag associated with LVD290_C VD, the voltage level detect for the high voltage Core 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 2.90 V point supply

Table 150. EPR_HV0 field descriptions (continued)

Field	Description
19 UVD15_DACCMP	UVD15_DACCMP flag This bit is the high-voltage status flag associated with UVD380_DACCMP VD, the voltage level detect for the high voltage DAC/CMP 3.80 V supply. It is asserted when the supply rises above its corresponding Voltage Detector 15 threshold, and clears when the supply falls below its corresponding UVD15 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 3.80 V point supply
18 UVD15_AS	UVD15_AS flag This bit is the high-voltage status flag associated with UVD380_AS VD, the voltage level detect for the high voltage SAR ADC 3.80 V supply. It is asserted when the supply rises above its corresponding Voltage Detector 15 threshold, and clears when the supply falls below its corresponding UVD15 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 3.80 V point supply
17 LVD11_OSC	LVD11_OSC flag This bit is the high-voltage status flag associated with LVD290_OSC VD, the voltage level detect for the high voltage OSC 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 2.90 V point supply
17 LVD11_DACCMP	LVD11_DACCMP flag This bit is the high-voltage status flag associated with LVD290_DACCMP VD, the voltage level detect for the high voltage DAC/CMP 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 2.90 V point supply

14.5.12 Reset event enable register (REE_HV0)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0044 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	REE11_AD	REE11_AS	REE11_IO0	REE11_IO1	0	REE11_FL	REE11_C	0	0	0	0	REE15_DACCMP	REE15_AS	REE11_OSC	REE11_DACCMP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 83. Reset Event Enable Register (REE_HV0)

Table 151. REE_HV0 field descriptions

Field	Description
30 REE11_AD	LVD11_AD reset enable This bit defines whether LVD290_AD VD, the voltage level detect for the high voltage assertion on the Sigma Delta ADC 2.90 V supply generates a system reset. The RES11_AD bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.
29 REE11_AS	LVD11_AS reset enable This bit defines whether LVD290_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 2.90 V supply generates a system reset. The RES11_AS bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.
28 REE11_IO0	LVD11_IO0 reset enable This bit defines whether LVD290_IO0 VD, the voltage level detect for the high voltage assertion on the IO Ring0 2.90 V supply generates a system reset. The RES11_IO0 bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.
27 REE11_IO1	LVD11_IO1 reset enable This bit defines whether LVD290_IO1 VD, the voltage level detect for the high voltage assertion on the IO Ring1 2.90 V supply generates a system reset. The RES11_IO1 bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.

Table 151. REE_HV0 field descriptions (continued)

Field	Description
25 REE11_FL	LVD11_FL reset enable This bit defines whether LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply of the cold point generates a system reset. The RES11_FL bit determines whether a system reset is enabled or not. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
24 REE11_C	LVD11_C reset enable This bit defines whether LVD290_C VD, the voltage level detect for the high voltage Core 2.90 V supply, of the cold point generates a system reset. The RES11_C bit determines whether a system reset is enabled or not. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
19 REE15_DACCMP	UVD15_DACCMP reset enable This bit defines whether UVD380_DACCMP VD, the voltage level detect for the high voltage DAC/CMP 3.80 V supply, of the cold point generates a system reset. The RES15_DACCMP bit determines whether a system reset is enabled or not. 0 Disabled. UVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. UVD assertion on the supply of the cold point causes system reset.
18 REE15_AS	UVD15_AS reset enable This bit defines whether UVD380_AS VD, the voltage level detect for the high voltage SAR ADC 3.80 V supply, of the cold point generates a system reset. The RES15_AS bit determines whether a system reset is enabled or not. 0 Disabled. UVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. UVD assertion on the supply of the cold point causes system reset.
17 REE11_OSC	LVD11_OSC reset enable This bit defines whether LVD290_OSC VD, the voltage level detect for the high voltage OSC 2.90 V supply, of the cold point generates a system reset. The RES11_OSC bit determines whether a system reset is enabled or not. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
17 REE11_DACCMP	LVD11_DACCMP reset enable This bit defines whether LVD290_DACCMP VD, the voltage level detect for the high voltage DAC/CMP 2.90 V supply, of the cold point generates a system reset. The RES11_DACCMP bit determines whether a system reset is enabled or not. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

14.5.13 Reset event select register (RES_HV0)

This Reset Event Select Register controls, in HIGH VOLTAGE 0 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0048 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	RES11_AD	RES11_AS	RES11_IO0	RES11_IO1	0	RES11_FL	RES11_C	0	0	0	0	RES15_DACCMP	RES15_AS	RES11_OSC	RES11_DACCMP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 84. Reset Event Select Register (RES_HV0)

Table 152. RES_HV0 field descriptions

Field	Description
30 RES11_AD	RES11_AD reset event select This bit defines whether an LVD assertion on the Sigma Delta ADC 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_ad bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
29 RES11_AS	RES11_AS reset event select This bit defines whether an LVD assertion on the SAR ADC 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_AS bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

Table 152. RES_HV0 field descriptions (continued)

Field	Description
28 RES11_IO0	RES11_IO0 reset event select This bit defines whether an LVD assertion on the IO Ring0 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_IO0 bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
27 RES11_IO1	RES11_IO1 reset event select This bit defines whether an LVD assertion on the IO Ring1 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_IO1 bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
25 RES11_FL	RES11_FL reset event select This bit defines whether an LVD assertion on the Flash 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_FL bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
24 RES11_C	RES11_C reset event select This bit defines whether an LVD assertion on the Core 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_C bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
19 RES15_DACCMP	RES15_DACCMP reset event select This bit defines whether an UVD assertion on the DAC/CMP 3.80 V supply of the cold point generates a destructive reset or a functional reset. The REE15_DACCMP bit determines whether the reset event is enabled. 0 Destructive Reset Generated. UVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. UVD assertion on the supply of the cold point causes a functional system reset.
18 RES15_AS	RES15_AS reset event select This bit defines whether an UVD assertion on the SAR ADC 3.80 V supply of the cold point generates a destructive reset or a functional reset. The REE15_AS bit determines whether the reset event is enabled. 0 Destructive Reset Generated. UVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. UVD assertion on the supply of the cold point causes a functional system reset.

Table 152. RES_HV0 field descriptions (continued)

Field	Description
17 RES11_OSC	RES11_OSC reset event select This bit defines whether an LVD assertion on the OSC 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_OSC bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
17 RES11_DACCMP	RES11_DACCMP reset event select This bit defines whether an LVD assertion on the DAC/CMP 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_DACCMP bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

14.5.14 Interrupt enable (IE_HV0)

This configuration register contains a set of HV0 interrupt Enable bits that correspond to each of the HV0 Event Pending Registers (EPR_HV0). These bits indicate whether an interrupt is enabled, waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-only monitor in the register (IE_G) at address 0x0088. The IE_HV0 bits can be read or written at any time.

Offset: 0x004C Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	LVD11IE_AD	LVD11IE_AS	LVD11IE_IO0	LVD11IE_IO1	0	LVD11IE_FL	LVD11IE_C	0	0	0	0	UVD15IE_DACCMP	UVD15IE_AS	LVD11IE_OSC	LVD11IE_DACCMP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 85. Interrupt Enable Register (IE_HV0)

Table 153. IE_HV0 field descriptions

Field	Description
30 LVD11IE_AD	LVD11IE_AD Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_AD VD, the voltage level detect for the high voltage assertion on the Sigma Delta ADC 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
29 LVD11IE_AS	LVD11IE_AS Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
28 LVD11IE_IO0	LVD11IE_IO0 Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_IO0 VD, the voltage level detect for the high voltage assertion on the IO Ring0 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
27 LVD11IE_IO1	LVD11IE_IO1 Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_IO1 VD, the voltage level detect for the high voltage assertion on the IO Ring1 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
25 LVD11IE_FL	LVD11IE_FL Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
24 LVD11IE_C	LVD11IE_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_C VD, the voltage level detect for the high voltage Core 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
19 UVD15IE_DACCMP	UVD15IE_DACCMP Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the UVD380_DACCMP VD, the voltage level detect for the high voltage DAC/CMP 3.80 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled

Table 153. IE_HV0 field descriptions (continued)

Field	Description
18 UVD15IE_AS	UVD15IE_AS Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the UVD380_AS VD, the voltage level detect for the high voltage SAR ADC 3.80 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
17 LVD11IE_OSC	LVD11IE_OSC Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_OSC VD, the voltage level detect for the high voltage OSC 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
17 LVD11IE_DACCMP	LVD11IE_DACCMP Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_DACCMP VD, the voltage level detect for the high voltage DAC/CMP 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled

14.5.15 FCCU event enable register (FEE_HV0)

This FCCU Event Enable Register HV0 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0050 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	FEE11_AD	FEE11_AS	FEE11_IO0	FEE11_IO1	0	FEE11_FL	FEE11_C	0	0	0	0	FEE15_DACCMP	FEE15_AS	FEE11_OSC	FEE11_DACCMP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 86. FCCU Event Enable Register (FEE_HV0)

Table 154. FEE_HV0 field descriptions

Field	Description
30 FEE11_AD	FEE11_AD FCCU event enable This bit defines whether a LVD290_AD VD, the voltage level detect for the high voltage assertion on the Sigma Delta ADC 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the ADC does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the ADC causes an FCCU event.
29 FEE11_AS	FEE11_AS FCCU event enable This bit defines whether a LVD290_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the ADC does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the ADC causes an FCCU event.
28 FEE11_IO0	FEE11_IO0 FCCU event enable This bit defines whether a LVD290_IO0 VD, the voltage level detect for the high voltage assertion on the IO Ring0 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the IO Ring0 does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the IO Ring0 causes an FCCU event.
27 FEE11_IO1	FEE11_IO1 FCCU event enable This bit defines whether a LVD290_IO1 VD, the voltage level detect for the high voltage assertion on the IO Ring1 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the IO Ring1 does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the IO Ring1 causes an FCCU event.
25 FEE11_FL	FEE11_FL FCCU event enable This bit defines whether a LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the Flash does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the Flash causes an FCCU event.
24 FEE11_C	FEE11_C FCCU event enable This bit defines whether a LVD290_C VD, the voltage level detect for the high voltage assertion on the Core 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the Core does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the Core causes an FCCU event.

Table 154. FEE_HV0 field descriptions (continued)

Field	Description
19 FEE15_DACCMP	FEE15_DACCMP FCCU event enable This bit defines whether a UVD380_DACCMP VD, the voltage level detect for the high voltage assertion on the DAC/CMP 3.80 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the DAC/CMP does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the DAC/CMP causes an FCCU event.
18 FEE15_AS	FEE15_AS FCCU event enable This bit defines whether a UVD380_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 3.80 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the SAR ADC does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the SAR ADC causes an FCCU event.
17 FEE11_OSC	FEE11_OSC FCCU event enable This bit defines whether a LVD290_OSC VD, the voltage level detect for the high voltage assertion on the OSC 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the OSC does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the OSC causes an FCCU event.
17 FEE11_DACCMP	FEE11_DACCMP FCCU event enable This bit defines whether a LVD290_DACCMP VD, the voltage level detect for the high voltage assertion on the DAC/CMP 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the DAC/CMP does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the DAC/CMP causes an FCCU event.

14.5.16 Supply gauge status register (GR_S)

This status register contains the gauge of the indicated supply. This indicates the present state of the voltage detect events for each of the supplies. Each of the voltage detect signals for a given voltage, these signals are combined to form a single bit, refer to *Table 155*.

Offset: 0x0080 Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	TEMP_2	TEMP_1	TEMP_0	VD3_PLL1	VD3_PLL0	VD3_DD	0	0	0	0	0	VD15_DACCMP	VD15_AS	VD11_OSC	VD11_DACCMP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	VD11_AD	VD11_AS	VD11_IO0	VD11_IO1	VD11_FL	VD11_C	VD7_RC	VD6_C	VD3_RC	0	VD3_FL	VD3_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 87. Supply Gauge Status Register (GR_S)

Table 155. GR_S field descriptions

Field	Description
30 TEMP_2	TEMP_2 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (150 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold. 0 Currently no occurrence 1 Temperature occurrence detected
29 TEMP_1	TEMP_1 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (135 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold. 0 Currently no occurrence 1 Temperature occurrence detected
28 TEMP_0	TEMP_0 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (-40 °C). It is asserted when the temperature falls below its corresponding threshold, and clears when the temperature is above its corresponding threshold. 0 Currently no occurrence 1 Temperature occurrence detected

Table 155. GR_S field descriptions (continued)

Field	Description
27 VD3_PLL1	VD3_PLL1 low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD119_PLL1 for the low voltage 1.19 V PLL1 supplies. It is asserted when any of the PLL1 supplies falls below the corresponding LVD threshold, and clears when all the PLL1 supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the low voltage 1.19 V supplies
26 VD3_PLL0	VD3_PLL0 low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD119_PLL0 for the low voltage 1.19 V PLL0 supplies. It is asserted when any of the PLL0 supplies falls below the corresponding LVD threshold, and clears when all the PLL0 supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the low voltage 1.19 V supplies
25 VD3_DD	VD3_DD low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD119_DD for the low voltage 1.19 V DLL supplies. It is asserted when any of the DLL supplies falls below the corresponding LVD threshold, and clears when all the DLL supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the low voltage 1.19 V supplies
19 VD15_DACCMP	VD15_DACCMP high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect UVD380_DACCMP for the high voltage 3.80 V DAC/CMP supplies. It is asserted when any of the DAC/CMP supplies rises above the corresponding UVD threshold, and clears when all the DAC/CMP supplies fall below the corresponding UVD threshold. All of the 3.80 V UVDs are combined to form this bit. 0 Currently no occurrence 1 UVD occurrence detected on the high voltage 3.80 V supplies
18 VD15_AS	VD15_AS high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect UVD380_AS for the high voltage 3.80 V SAR ADC supplies. It is asserted when any of the SAR ADC supplies rises above the corresponding UVD threshold, and clears when all the SAR ADC supplies fall below the corresponding UVD threshold. All of the 3.80 V UVDs are combined to form this bit. 0 Currently no occurrence 1 UVD occurrence detected on the high voltage 3.80 V supplies

Table 155. GR_S field descriptions (continued)

Field	Description
17 VD11_OSC	VD11_OSC high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect LVD290_OSC for the high voltage 2.90 V OSC supplies. It is asserted when any of the OSC supplies falls below the corresponding LVD threshold, and clears when all the OSC supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
16 VD11_DACCMP	VD11_DACCMP high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect LVD290_DACCMP for the high voltage 2.90 V DAC/CMP supplies. It is asserted when any of the DAC/CMP supplies falls below the corresponding LVD threshold, and clears when all the DAC/CMP supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
11 VD11_AD	VD11_AD high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect LVD290_AD for the high voltage 2.90 V Sigma Delta ADC supplies. It is asserted when any of the ADC supplies falls below the corresponding LVD threshold, and clears when all the ADC supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
10 VD11_AS	VD11_AS high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect LVD290_AD for the high voltage 2.90 V SAR ADC supplies. It is asserted when any of the ADC supplies falls below the corresponding LVD threshold, and clears when all the ADC supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
9 VD11_IO0	VD11_IO0 high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect LVD290_IO0 for the high voltage 2.90 V IO Ring0 supplies. It is asserted when any of the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies

Table 155. GR_S field descriptions (continued)

Field	Description
8 VD11_IO1	VD11_IO1 high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect LVD290_IO1 for the high voltage 2.90 V IO Ring1 supplies. It is asserted when any of the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
7 VD11_FL	VD11_FL high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect LVD290_FL for the high voltage 2.90 V Flash supplies. It is asserted when any of the Flash supplies falls below the corresponding LVD threshold, and clears when all the Flash supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
6 VD11_C	VD11_C high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect LVD290_C for the high voltage 2.90 V Core supplies. It is asserted when any of the Core supplies falls below the corresponding LVD threshold, and clears when all the Core supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
5 VD7_RC	VD7_RC low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect UVD145_RC for the low voltage 1.45 V RCOSC supplies. It is asserted when any of the RCOSC supplies rises above the corresponding UVD threshold, and clears when all the RCOSC supplies fall below the corresponding UVD threshold. All the 1.45 V UVDs are combined to form this bit. 0 Currently no occurrence 1 UVD occurrence detected on the low voltage 1.45 V supply
4 VD6_C	VD6_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect HVD140_C for the low voltage 1.40 V Core supplies. It is asserted when any of the Core supplies rises above the corresponding HVD threshold, and clears when all the Core supplies fall below the corresponding HVD threshold. All the 1.40 V HVDs are combined to form this bit. 0 Currently no occurrence 1 HVD occurrence detected on the low voltage 1.40 V supply

Table 155. GR_S field descriptions (continued)

Field	Description
3 VD3_RC	VD3_RC low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD119_RC for the low voltage 1.19 V RCOSC supplies. It is asserted when any of the RCOSC supplies falls below the corresponding LVD threshold, and clears when all the RCOSC supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the low voltage 1.19 V supplies
1 VD3_FL	VD3_FL low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD119_FL for the low voltage 1.19 V Flash supplies. It is asserted when any of the Flash supplies falls below the corresponding LVD threshold, and clears when all the Flash supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the low voltage 1.19 V supplies
0 VD3_C	VD3_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD119_C for the low voltage 1.19 V Core supplies. It is asserted when any of the Core supplies falls below the corresponding LVD threshold, and clears when all the Core supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the low voltage 1.19 V supplies

14.5.17 Pending gauge status register (GR_P)

This configuration register contains a mirror of the contents of the Event Pending Registers, but in a format that includes one bit for each voltage detect signal. These bits indicate the state of the voltage detect events including whether a voltage level event has occurred at any time in the past but has not been cleared (via the EPR register). These bits are read only, and they are cleared by writing a one to the corresponding EPR register.

The Pending registers are only set on an active edge of an event, so a transition must be seen before they become set.

The raw analog block voltage detect signals are also synchronized to the bus clock before being used for the pending registers.

Offset: 0x0084 Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	VD3_PLL1	VD3_PLL0	VD3_DD	0	0	0	0	0	VD15_DACCMP	VD15_AS	VD11_OSC	VD11_DACCMP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	VD11_AD	VD11_AS	VD11_IO0	VD11_IO1	VD11_FL	VD11_C	VD7_RC	VD6_C	VD3_RC	0	VD3_FL	VD3_C	0	IRQ_ST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 88. Pending Gauge Status Register (GR_P)

Table 156. GR_P field descriptions

Field	Description
27 VD3_PLL1	VD3_PLL1 Pending Monitor. Read-only register.
26 VD3_PLL0	VD3_PLL0 Pending Monitor. Read-only register.
25 VD3_DD	VD3_DD Pending Monitor. Read-only register.
19 VD15_DACCMP	VD15_DACCMP Pending Monitor. Read-only register.
18 VD15_AS	VD15_AS Pending Monitor. Read-only register.
17 VD11_OSC	VD11_OSC Pending Monitor. Read-only register.
16 VD11_DACCMP	VD11_DACCMP Pending Monitor. Read-only register.
14 VD11_AD	VD11_AD Pending Monitor. Read-only register.
13 VD11_AS	VD11_AS Pending Monitor. Read-only register.
12 VD11_IO0	VD11_IO0 Pending Monitor. Read-only register.
11 VD11_IO1	VD11_IO1 Pending Monitor. Read-only register.

Table 156. GR_P field descriptions (continued)

Field	Description
10 VD11_FL	VD11_FL Pending Monitor. Read-only register.
9 VD11_C	VD11_C Pending Monitor. Read-only register.
8 VD7_RC	VD7_RC Pending Monitor. Read-only register.
7 VD6_C	VD6_C Pending Monitor. Read-only register.
6 VD3_RC	VD3_RC Pending Monitor. Read-only register.
4 VD3_FL	VD3_FL Pending Monitor. Read-only register.
3 VD3_C	VD3_C Pending Monitor. Read-only register.
1 IRQ_ST	IRQ_ST user BIST Pending Monitor. Read-only register.

14.5.18 Global interrupt enable register (IE_G)

This configuration register contains a set of Interrupt Enable bits that correspond to each of the Event Pending Registers (EPR_XX), but in a format similar to the Pending Gauge Status register (GR_P). These bits indicate whether an interrupt is enabled. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are enabled via a write of ‘1’ to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.

Offset: 0x0088 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IE_EN	0	0	0	VD3IE_PLL1	VVD3IE_PLL0	VD3IE_DD	0	0	0	0	0	VD15IE_DACCMP	VD15IE_AS	VD11IE_OSC	VD11IE_DACCMP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	VD11IE_AD	VD11IE_AS	VD11IE_IO0	VD11IE_IO1	VD11IE_FL	VD11IE_C	VD7IE_RC	VD6IE_C	VD3IE_RC	0	VD3IE_FL	VD3IE_C	0	IRQ_EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 89. Global Interrupt Enable Register (IE_G)

Table 157. IE_G field descriptions

Field	Description
31 IE_EN	IE_EN. Interrupt Enable. This bit permits Interrupt Enable bit to be written. 0 No interrupt enables can be written. 1 Any interrupt enable can be written.
27 VD3IE_PLL1	VD3IE_PLL1 enable interrupt monitor. Read-only register.
26 VD3IE_PLL0	VD3IE_PLL0 enable interrupt monitor. Read-only register.
25 VD3IE_DD	VD3IE_DD enable interrupt monitor. Read-only register.
19 VD15IE_DACCMP	VD15IE_DACCMP enable interrupt monitor. Read-only register.
18 VD15IE_AS	VD15IE_AS enable interrupt monitor. Read-only register.
17 VD11IE_OSC	VD11IE_OSC enable interrupt monitor. Read-only register.
16 VD11IE_DACCMP	VD11IE_DACCMP enable interrupt monitor. Read-only register.
14 VD11IE_AD	VDIE11_AD enable interrupt monitor. Read-only register.
13 VD11IE_AS	VDIE11_AS enable interrupt monitor. Read-only register.
12 VD11IE_IO0	VDIE11_IO0 enable interrupt monitor. Read-only register.
11 VD11IE_IO1	VDIE11_IO1 enable interrupt monitor. Read-only register.
10 VD11IE_FL	VDIE11_FL enable interrupt monitor. Read-only register.
9 VD11IE_C	VDIE11_C enable interrupt monitor. Read-only register.
8 VD7IE_RC	VD7IE_RC enable interrupt monitor. Read-only register.
7 VD6IE_C	VDIE6_C enable interrupt monitor. Read-only register.
6 VD3IE_RC	VD3IE_RC enable interrupt monitor. Read-only register.
4 VD3IE_FL	VDIE3_FL enable interrupt monitor. Read-only register.

Table 157. IE_G field descriptions (continued)

Field	Description
3 VD3IE_C	VDIE3_C enable interrupt monitor. Read-only register.
1 IRQ_EN	IRQ_EN user BIST enable interrupt monitor. Read-only register.

14.5.19 HPREG SMPS SELECT STATUS (HPREG_SMPS_SEL_S)

This register shows the status of SMPS.

Offset: 0x0244 Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	SMPS_ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 90. HPREG SMPS SELECT STATUS register (HPREG_SMPS_SEL_S)

Table 158. HPREG_SMPS_SEL_S field descriptions

Field	Description
0 SMPS_ENB	SMPS enable/disable status 0 SMPS enabled 1 SMPS disabled

14.5.20 Event pending register (EPR_TD)

This Event Pending Register indicates the present and past state of the temperature sensor signals. If the temperature event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a ‘1’ must be written to the flag bit that needs to be cleared.

Offset: 0x0300 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 91. Event Pending Register (EPR_TD)

Table 160. EPR_TD field descriptions

Field	Description
2 TEMP_2	TEMP_2 flag. This bit is the temperature status flag associated with the temperature for the temperature sensor point (150 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a '1' is written to this bit location. If the IETD_2 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP_2] is also asserted, a system reset is generated, which clears IETD_2 and negate the interrupt request. If REE_TD[TEMP_2] is asserted and a destructive reset is selected (via RES_TD[TEMP_2] being 0), then a destructive reset is generated. If RES_TD[TEMP_2] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected
1 TEMP_1	TEMP_1 flag. This bit is the temperature status flag associated with the temperature for the temperature sensor point (135 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a '1' is written to this bit location. If the IETD_1 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP_1] is also asserted, a system reset is generated, which clears IETD_1 and negate the interrupt request. If REE_TD[TEMP_1] is asserted and a destructive reset is selected (via RES_TD[TEMP_1] being 0), then a destructive reset is generated. If RES_TD[TEMP_1] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected

Table 160. EPR_TD field descriptions (continued)

Field	Description
0 TEMP_0	TEMP_0 flag. This bit is the temperature status flag associated with the temperature for the *cold* temperature sensor point. It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a '1' is written to this bit location. If the IETD_0 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP_0] is also asserted, a system reset is generated, which clears IETD_0 and negate the interrupt request. If REE_TD[TEMP_0] is asserted and a destructive reset is selected (via RES_TD[TEMP_0] being 0), then a destructive reset is generated. If RES_TD[TEMP_0] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected

14.5.21 Reset event enable register (REE_TD)

This Reset Event Enable Register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits also get cleared by the destructive reset.

Offset: 0x0304 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 92. Reset Event Enable Register (REE_TD)

Table 161. REE_TD field descriptions

Field	Description
2 TEMP_2	TEMP_2 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_2] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
1 TEMP_1	TEMP_1 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_1] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
0 TEMP_0	TEMP_0 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.

14.5.22 Reset event select register (RES_TD)

This Reset Event Select Register controls whether the temperature sensor detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the temperature sensor event causes either a destructive or functional reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits also get cleared by the destructive reset.

Offset: 0x0308 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 93. Reset Event Select Register (RES_TD)

Table 162. RES_TD field descriptions

Field	Description
2 TEMP_2	TEMP_2 reset event select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_2] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
1 TEMP_1	TEMP_1 reset event select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_1] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
0 TEMP_0	TEMP_0 reset event select. This bit defines whether an temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_0] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.

14.5.23 Temperature sensor configuration register (CTL_TD)

The Temperature Sensor Configuration Registers (CTL_TD) contains two enables that must both be enabled in order for the Temperature Sensor to be operable. It also includes two 5-bit signed trim adjustment register fields that are used to modify both TS trim values (one bus for over temperature and one bus for under temperature).

Offset: 0x030C Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	PMC_AOUT_EN	TRIM_ADJ_OVER1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TS2IE	TS1IE	TS0IE	0	TRIM_ADJ_OVER				0	0	TRIM_ADJ_UNDER				AMUX_CTAT_TS_CTRL	ADC_AOUT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 94. Temperature Sensor Configuration Register (CTL_TD)

Table 163. CTL_TD field descriptions

Field	Description
20 PMC_AOUT_EN	PMC_AOUT_EN. Temperature Sensor Enable for PMU. Refer to the Temperature Sensor chapter for details.
19:16 TRIM_ADJ_OVER1	TRIM_ADJ_OVER1[3:0]. Customer adjustable over trim register for PMU. These bits are a signed binary number that is added to the over temperature trim (NT_TD1) value. The TRIM value is protected from overflows and underflows due to this addition. Here MSB bit is sign bit and third bit is tied to zero, the actual trim values are on 2:0 bits. Refer to the Temperature Sensor chapter for details.
15 TS2IE	TS2IE Temperature Sensor input 2 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
14 TS1IE	TS1IE Temperature Sensor input 1 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
13 TS0IE	TS0IE Temperature Sensor input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
11:8 TRIM_ADJ_OVER	TRIM_ADJ_OVER[3:0]. Customer adjustable over trim register for ADC. These bits are a signed binary number that is added to the over temperature trim (NT_TD2) value. The TRIM value is protected from overflows and underflows due to this addition. Here MSB bit is sign bit and third bit is tied to zero, the actual trim values are on 2:0 bits. Refer to the Temperature Sensor chapter for details.
5:2 TRIM_ADJ_UNDER	TRIM_ADJ_UNDER[3:0]. Customer adjustable under trim register. These bits are a signed binary number that is added to the Under temperature trim value (NT_TD0). The TRIM value is protected from overflows and underflows due to this addition. Here MSB bit is sign bit and third bit is tied to zero, the actual trim values are on 2:0 bits. Refer to the Temperature Sensor chapter for details.
1 AMUX_CTAT_TS_CTRL	AMUX_CTAT_TS_CTRL. Analog mux enable. Refer to the Temperature Sensor chapter for details.
0 ADC_AOUT_EN	ADC_AOUT_EN. Temperature Sensor Enable for ADC. Refer to the Temperature Sensor chapter for details.

14.5.24 Temperature sensor FCCU event enable register (FEE_TD)

This Temperature Sensor FCCU Event Enable Register controls whether the temperature sensor detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit (EPR_TD) is enabled, the temperature sensor detect assertion causes an FCCU event to be generated when the selected temperature exceeds the desired value.

Offset: 0x0318 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE_TS2	FEE_TS1	FEE_TS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 95. Temperature Sensor FCCU Event Enable Register (FEE_TD)

Table 164. Temperature Sensor FEE_TD field descriptions

Field	Description
2 FEE_TS2	FEE_TS2 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.
1 FEE_TS1	FEE_TS1 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.
0 FEE_TS0	FEE_TS0 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.

14.5.25 BIST mask status register (BIST_MASK_STATUS)

This status register contains a monitor of the BIST MASK signal.

Offset: 0x03C8 Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	BIST_MASK									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BIST_MASK															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 96. BIST Mask Status register (BIST_MASK_STATUS)

Table 165. BIST_MASK_STATUS field descriptions

Field	Description
25:0 BIST_MASK	BIST mask status Bit 25: mvd114b_c_1p2d_mask status Bit 24: mvd114b_fa_1p2d_mask status Bit 23: lvd119b_pll0_1p2d_mask status Bit 22: lvd119b_c_1p2d_mask status Bit 21: lvd119b_fl_1p2d_mask status Bit 20: lvd119b_pll1_1p2d_mask status Bit 19: lvd119b_dd_1p2d_mask status Bit 18: lvd119b_rc_1p2d_mask status Bit 17: mvd270b_c_main_1p2d_mask status Bit 16: mvd270b_fl_1p2d_mask status Bit 15: lvd290b_c_1p2d_mask status Bit 14: lvd290b_fl_1p2d_mask status Bit 13: lvd290b_ad_1p2d_mask status Bit 12: lvd290b_as_1p2d_mask status Bit 11: lvd290b_io1_1p2d_mask status Bit 10: lvd290b_io0_1p2d_mask status Bit 9: lvd290b_osc_1p2d_mask status Bit 7: lvd290b_daccmp_1p2d_mask status Bit 7: hvd140b_c_1p2d_mask status Bit 6: uvd145b_c_1p2d_mask status Bit 5: uvd145b_rc_1p2d_mask status Bit 4: uvd380b_c_1p2d_mask status Bit 3: uvd380b_fl_1p2d_mask status Bit 2: uvd380b_as_1p2d_mask status Bit 1: uvd380b_io0_1p2d_mask status Bit 0: uvd380b_daccmp_1p2d_mask status

14.5.26 User BIST flags phase1 register (BIST_FLAGS_PHASE1)

This configuration register contains a monitor of the flags phase1 coming from the User BIST sub-block.

Offset: 0x03CC Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	FLAGS_P1									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLAGS_P1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 97. User BIST Flags Phase1 register (BIST_FLAGS_PHASE1)

Table 166. BIST_FLAGS_PHASE1 field descriptions

Field	Description
25:0 FLAGS_P1	The 26 bits of field FLAGS_P1 are the phase one flags of the User BIST that indicate the status of monitor when it is tripped: Bit 25 - UVD380B_DACCMP Bit 24 - UVD380B_IO0 Bit 23 - UVD380B_AS Bit 22 - UVD380B_FL Bit 21 - UVD380B_C Bit 20 - UVD145B_RC Bit 19 - UVD145B_C Bit 18 - HVD140B_C Bit 17 -LVD290B_DACCMP Bit 16 - LVD290B_OSC Bit 15 - LVD290B_IO0 Bit 14 - LVD290B_IO1 Bit 13 - LVD290B_AS Bit 12 - LVD290B_AD Bit 11 - LVD290B_FL Bit 10 - LVD290B_C Bit 9 - MVD270B_FL Bit 8 - MVD270B_C_MAIN Bit 7 - LVD119B_RC Bit 6 - LVD119B_DD Bit 5 - LVD119B_PLL1 Bit 4 - LVD119B_FL Bit 3 - LVD119B_C Bit 2 - LVD119B_PLL0 Bit 1 - MVD114B_FA Bit 0 - MVD114B_C

14.5.27 User BIST flags phase2 register (BIST_FLAGS_PHASE2)

This configuration register contains a monitor of the flags phase2 coming from the User BIST sub-block.

Offset: 0x03D0 Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	FLAGS_P2									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLAGS_P2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 98. User BIST Flags Phase2 register (BIST_FLAGS_PHASE2)

Table 167. BIST_FLAGS_PHASE2 field descriptions

Field	Description
25:0 FLAGS_P2	The 26 bits of field FLAGS_P2 are the phase two flags of the User BIST that indicate the status of monitor when it is lifted: Bit 25 - UVD380B_DACCMP Bit 24 - UVD380B_IO0 Bit 23 - UVD380B_AS Bit 22 - UVD380B_FL Bit 21 - UVD380B_C Bit 20 - UVD145B_RC Bit 19 - UVD145B_C Bit 18 - HVD140B_C Bit 17 -LVD290B_DACCMP Bit 16 - LVD290B_OSC Bit 15 - LVD290B_IO0 Bit 14 - LVD290B_IO1 Bit 13 - LVD290B_AS Bit 12 - LVD290B_AD Bit 11 - LVD290B_FL Bit 10 - LVD290B_C Bit 9 - MVD270B_FL Bit 8 - MVD270B_C_MAIN Bit 7 - LVD119B_RC Bit 6 - LVD119B_DD Bit 5 - LVD119B_PLL1 Bit 4 - LVD119B_FL Bit 3 - LVD119B_C Bit 2 - LVD119B_PLL0 Bit 1 - MVD114B_FA Bit 0 - MVD114B_C

14.5.28 User BIST control register (BIST_CTRL)

The User BIST Control register (BIST_CTRL) contains both control bits and status bits to manage the User BIST.

Offset: 0x03D4 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	IRQST	0	0	0	IRQEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	NCFST				NCFEN	0	STATUS			0	0	0	START
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 99. User BIST Control register (BIST_CTRL)

Table 168. BIST_CTRL field descriptions

Field	Description
20 IRQST	IRQST is the interrupt request pending status bit to generate a User BIST Interrupt. It is a R/W0 register. That means Set by HW and Cleared by SW. The destructive reset resets this bit. 0 No interrupt occurred 1 An interrupt occurred
16 IRQEN	IRQEN is the interrupt request enable bit to permit a User BIST Interrupt. It is a R/W register. The destructive reset resets this bit. 0 Interrupt is disabled 1 Interrupt enabled
12 NCFST	NCFST is the Not Critical Fault bit. The assertion of this bit generates FCCU event once user BIST fails. It is set by HW and Cleared by SW. The destructive reset resets this bit. 0 No User BIST NCF occurred 1 A User BIST NCF occurred
8 NCFEN	NCFEN is the Not Critical Fault enable bit to permit a User BIST Not Critical Fault and Fault indication is sent to FCCU. It is a R/W register. The destructive reset resets these bits. 0 No User BIST NCF enabled 1 A User BIST NCF enabled
6:4 STATUS	STATUS register consists of 3 read-only bits, that according to the value has its own meaning. 000 BIST_IDLE 001 BIST_RUN 010 BIST_PASSED 011 BIST_FAILED 100 BIST_ABORT 101 Reserved 110 Reserved 111 Reserved
0 START	START User BIST bit. If set by SW a 1 is present in the START bit for a System clock cycle only, then the START bit register is clear by HW. The destructive reset resets this bit. 0 No start pulse. 1 A start pulse occurs of a 1 System clock cycle duration

14.5.29 User BIST Time1 and Time0 register (BIST_TIME10)

The User BIST Time1 and Time0 Register (BIST_TIME10) contains two time numbers Time1 and Time0 giving the System clock cycles number required by the User BIST timer1 and timer0 (refer to *Section 14.9: Voltage monitor BIST programming* for more information regarding the User BIST timer registers).

Offset: 0x03D8 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	TIME_1											
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	TIME_0											
W																
Reset	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1

Figure 100. User BIST Time1 and Time0 register (BIST_TIME10)

Table 169. BIST_TIME10 field descriptions

Field	Description
28:16 TIME_1	TIME_1 is the system clock cycles number required by the TIMER 1 in the User BIST management. The destructive reset resets these bits.
12:0 TIME_0	TIME_0 is the system clock cycles number required by the TIMER 0 in the User BIST management. The destructive reset resets these bits.

14.5.30 User BIST Time3 and Time2 register (BIST_TIME32)

The User BIST Time3 and Time2 Register (BIST_TIME32) contains two time numbers Time3 and Time2 giving the system clock cycles number required by the User BIST timer3 and timer2 (refer to *Section 14.9: Voltage monitor BIST programming* for more information regarding the User BIST timer registers).

Offset: 0x03DC Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	TIME_3											
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	TIME_2											
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 101. User BIST Time3 and Time2 register (BIST_TIME32)

Table 170. BIST_TIME32 field descriptions

Field	Description
28:16 TIME_3	TIME_3 is the system clock cycles number required by the TIMER 3 in the User BIST management. The destructive reset resets these bits.
12:0 TIME_2	TIME_2 is the system clock cycles number required by the TIMER 2 in the User BIST management. The destructive reset resets these bits.

14.5.31 User BIST Time6 and Time5 register (BIST_TIME65)

The User BIST Time6 and Time5 Register (BIST_TIME65) contains two time numbers Time6 and Time5 giving the system clock cycles number required by the User BIST timer6 and timer5 (refer to *Section 14.9: Voltage monitor BIST programming* for more information regarding the User BIST timer registers).

Offset: 0x03E0 Access: User read/write, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	TIME_6											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	TIME_5											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Figure 102. User BIST Time6 and Time5 register (BIST_TIME65)

Table 171. BIST_TIME65 field descriptions

Field	Description
28:16 TIME_6	TIME_6 is the system clock cycles number required by the TIMER 6 in the User BIST management. The destructive reset resets these bits.
12:0 TIME_5	TIME_5 is the system clock cycles number required by the TIMER 5 in the User BIST management. The destructive reset resets these bits.

14.5.32 User BIST VD under test monitor register (BIST_DEBUG)

This VD Under Test Monitor register contains a monitor of the Voltage Detector Under Test of the USER BIST sub-block.

Note: The BIST_DEBUG register can only be read while BIST is running. Once BIST is finished the status can be obtained by checking the BIST_FLAGS_PHASE1 and BIST_FLAGS_PHASE2 registers.

Offset: 0x03E4 Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	VD_MON					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 103. User BIST VD Under Test Monitor register (BIST_DEBUG)

Table 172. BIST_DEBUG field description

Field	Description
5:0 VD_MON	The six read only bits of field VD_MON give the current VD Under Test of the User BIST unit, among 26 VDs available, according to the following values: 0x1E: MVD114_C 0x1D: MVD114_FA 0x1C: LVD119_PLL0 0x1B: LVD119_C 0x1A: LVD119_FL 0x16: LVD119_DD 0x15: MVD270_C 0x14: MVD270_FL 0x13: LVD290_C 0x12: LVD290_FL 0x11: LVD290_AD 0x10: LVD290_AS 0x0F: LVD290_IO1 0x0E: LVD290_IO0 0x06: UVD145_RC 0x19: HVD140_C 0x18: UVD145_C 0x17: LVD119_PLL1 0x0D: LVD290_OSC 0x0C: LVD290_DACCMP 0x0B: UVD380_DACCMP 0x05: UVD380_C 0x04: UVD380_FL 0x03: UVD380_AS 0x02: UVD380_IO0 0x01: LVD119_RC 0x00: No VD under Test

14.6 Temperature Sensor Interface Logic

The Temperature Sensor interface consists of an analog enable, a set of mode registers, and an overtemp threshold select control.

For more information regarding the Temperature Sensor function, refer to the *Chapter 33: Temperature sensor*.

14.7 Power on reset (RCC phase gates)

The PMU digital logic interfaces to the Reset Generation Module in order to indicate the type and conditions of the resets from the PMU and Temperature Sensor Analog blocks.

There are two signals for both the PMU and Temp Sensor blocks; one is used to indicate a destructive reset, and one is used to indicate a functional reset.

The PMU digital logic also indicates when the RCC allows transition to the next POR phase.

The digital PMU also uses signals from the RCC to indicate when some of the reset signals have to be masked during the POR sequence.

During the POR phase 3 exit, the PMU digital logic uses all the enabled Voltage Detects to determine if it is correct to exit from Phase 3. Once all the enabled LVDs and HVDs have become deasserted and the time from the LVD circuits to adjust to new trim values has elapsed, then the signal is sent to the RCC so that the correct voltage levels have been achieved to exit from Phase 3.

14.8 Flash interface (DCF client usage)

The PMU digital is updated during POR from the Flash block via a DCF client. This load is used for several pieces of data, but most of the data is trim values being used for analog modules (PMU, Temperature Sensor, and ADC Bandgap).

These trim bits that are loaded from the Flash are not readable via the IP bus. These trim values loaded via the DCF client are not compacted (they are written individually, and not several at a time).

A DCF client is used to interface to the Flash for these early data accesses. This client also performs triple voting to keep the data safe from stray bit flips.

Another set of data that is loaded during the DCF single read process are updated values for the REE and RES bits. Refer to:

- *Section 14.5.2: Reset event enable register (REE_LV0)*,
- *Section 14.5.3: Reset event select register (RES_LV0)*,
- *Section 14.5.7: Reset event enable register (REE_LV1)*,
- *Section 14.5.8: Reset event select register (RES_LV1)*,
- *Section 14.5.12: Reset event enable register (REE_HV0)* and
- *Section 14.5.13: Reset event select register (RES_HV0)* for more information regarding the REE/RES bits load from Flash.

If the REE bit is loaded via the DCF as set (enabled), then it is not possible to clear this REE bit. The reset value of the REE DCF bits are all low (thus to allow the bits to be rewritable) and the same is the reset value of the REE register.

The monitors for which reactions are enabled are programmed in the corresponding bit flag of Reset Event Enable (REE_XX) register as “1” (reset enabled) and in the Reset Event Select (RES_XX) register as “0” (destructive reset selected).

The monitors for which reactions are not enabled are programmed in the corresponding bit flag of Reset Event Enable (REE_XX) register as “0” (reset disabled) and in the Reset Event Select (RES_XX) register as “1” (functional reset selected).

The REE DCF bits loaded are compacted into a single DCF write (they do not match the IPS Bus register bit locations). The REE DCF loaded bits do not have the same limitation that they can only be written once and any further writes must be to the same value. The REE DCF bits can be rewritten to a new value during a new reset event.

When the REE DCF bits are updated, the REE and RES register bits are updated at the same time to the new value.

All of the DCF clients are reset via the POR reset signal.

For more information regarding the DCF records, refer to the DCF Records chapter.

The trim data is read out of the Flash's differential read space (slower access, but less issues with voltage variances). Once the differential reads are completed, the Flash indicates this to the PMU digital via the SSCM module.

Once the PMU trim values have been read, the analog circuit has been given time to update to the new trim values, and the LVDs have all deasserted, a signal is asserted to the Flash to indicate that the voltage is high enough for faster single access reads.

All of the LVD signals must indicate that a valid voltage has been reached before the voltage indication is sent to the Flash.

14.9 Voltage monitor BIST programming

The Built-In Self Test (BIST) is a software initiated testing procedure used to verify all the voltage monitors.

The application has to program the BIST_TIMExx registers according to the following timings with respect to the configured PCLK1:

- BIST_TIME10[Time_0] = 40 µs.
- BIST_TIME10[Time_1] = 50 µs.
- BIST_TIME32[Time_2] = 50 µs.
- BIST_TIME32[Time_3] = 50 µs.
- BIST_TIME65[Time_5] = 200 ns.
- BIST_TIME65[Time_6] = 200 ns.

In case the aforementioned timings are not applied, some voltage monitor BIST could trigger fake faults.

The user can run a BIST mode by setting the BIST_CTRL[START] register bit. The following voltage detectors are then serially tested:

- MVD114_C
- MVD114_FA
- LVD119_PLL0
- LVD119_C
- LVD119_FL
- LVD119_PLL1
- LVD119_DD
- LVD119_RC
- MVD270_C
- MVD270_FL
- LVD290_C
- LVD290_FL
- LVD290_AD
- LVD290_AS
- LVD290_IO1
- LVD290_IO0
- LVD290_OSC
- LVD290_DACCMP
- HVD140_C
- UVD145_C
- UVD145_RC
- UVD380_C
- UVD380_FL
- UVD380_AS
- UVD380_IO0
- UVD380_DACCMP

Interrupt can be enabled by programming BIST_CTRL[IRQEN] and interrupt status is checked with BIST_CTRL[IRQST] on completion of BIST sequence.

FCCU event can be enabled by programming BIST_CTRL[NCFEN]. The BIST_CTRL[NCFST] status indicates BIST Fail on completion of BIST sequence.

BIST_DEBUG[VD_MON] register bits allow to know which voltage detector is under test.

The overall status of BIST is provided by the BIST_CTRL[STATUS] register bits.

The status of individual monitors is checked in flag registers BIST_FLAGS_PHASE1 (pass is all zero) and BIST_FLAGS_PHASE2 (pass is all one).

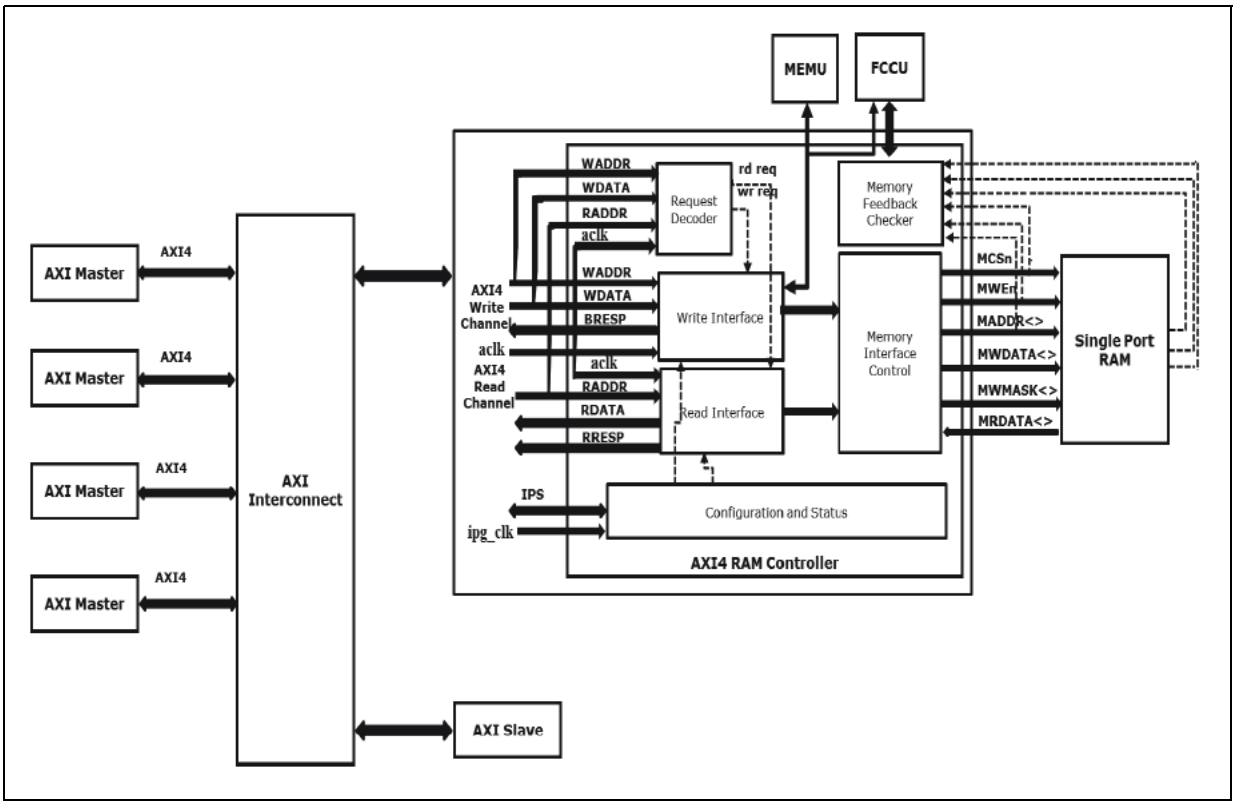
15 Platform RAM controller AXI (PRAMC_AXI)

15.1 Introduction

This section provides an overview of the Platform RAM Controller AXI. The module acts as an interface between the system bus (AMBA AXI4) and the integrated System RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

Figure 104 below is a simplified block diagram depicting the position of the PRAM controller AXI within a typical platform architecture:

Figure 104. Platform block diagram



The RAM controller supports one AXI interface configurable to 64-bit/128-bit/256-bit and a RAM array interface configurable to 64-bit/128-bit/256-bit.

The block consists of two main modules, one of them being the AXI4 read interface and the other AXI4 write interface. Both the read and write interfaces support all AXI4 burst types and sizes.

15.2 Features

The following list summarizes the key features of the RAM controller:

- The RAM controller adheres to the AMBA AXI4 Interface standard.

It includes the following features with few exceptions:

– Support for all AXI4 burst types and sizes:

  - Handling of FIXED bursts is configurable in the design.
  - Support INCR burst sizes up to 256 data transfers and WRAP bursts of 2, 4, 8 and 16 beats.
- Support for narrow width transactions where data transfers are on different byte lanes for each beat of the burst.
- Support for exclusives using the AXI lock attribute.
- No support for user, prot, and qos AXI attributes.
- No support for cachability or bufferability.
- No support for outstanding transactions. It accepts a single address for each of the read/write channels.
- No support for reordering. Transactions are executed in order regardless of transaction ID.
- It has the attributes listed in the following table:

Table 173. PRAMC_AXI slave attributes

Slave attribute	Description	Value
Write acceptance capability	Max. number of active writes that a slave can accept	1
Read acceptance capability	Max. number of active reads that a slave can accept	1
Write interleave depth	No. of active writes for which slave can receive data	1
Read data reorder depth	No. of active reads for which slave can transmit data	1

- The RAM Controller provides a single port memory interface for SRAM/ROM with configurable RAM data width (64-bit, 128-bit, 256-bit) which is the same as AXI Bus Width.
- The RAM Controller provides support for read-modify-write operation to support memory writes with sparse byte strobes and any access less than ECC granularity.
- The RAM Controller provides support for Exclusives for multiple processing elements.
- The RAM Controller assigns priority to the read channel when presented with simultaneous assertions of read address phase, write address phase and write data phase.
- The RAM Controller provides configurability for wait-state on access to RAM array to target higher memory access times. It is zero or one state programmable.
- The RAM Controller provides punch-through clock generation capability to enable memories to run at half the system frequency. This can be configured by adding wait state on read as well as write.

- The RAM Controller provides support for Memory Reconstruction to enable reconstruction of memory image for debug and trace purposes.
- The RAM Controller provides support for IPS interface which provides a memory mapped access to RAMC's programming model:
  - Programming model can be referenced only using 32-bit access. Any accesses using different access sizes to reserve addresses are terminated with an IPS error.
  - Programming model access is allowed only in supervisor mode.
- The RAM Controller provides 64-bit ECC granularity. The ECC scheme is aligned to the Cortex®-M7.
- The RAM Controller extracts the check-bits from the original check-bits in case of a read-modify-operation to provide end to end coverage.
- The RAM Controller performs an EDC check on the data read from memory to report any memory error to MEMU2.

15.3 Memory map and register description

The Platform RAM controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model can only be referenced using a 32-bit access. Attempted references using different access sizes or to reserved addresses generate an IPS error termination. The programming model can only be accessed in supervisor mode.

Attempted updates to the programming model while the RAM controller module is in the midst of an operation result in undetermined behavior. The software architecture must avoid this scenario.

15.3.1 Memory map

Table 174. PRAMC_AXI memory map

Address offset	Register	Location
0x0000	Platform RAM Controller Configuration Register (CR)	*Section 15.3.2.1*
0x0004–0x0007	Reserved	

15.3.2 Register definitions

15.3.2.1 Platform RAM controller Configuration Register (CR)

The platform RAM controller Configuration Register (CR) is used to specify operation of the RAM Controller.

Offset: 0x0000 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	WWS_EN	MEMACC_WAIT	RWS_EN	FIXED_BURST_EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 105. Platform RAM controller Configuration Register (CR)

Table 175. CR field descriptions

Field	Description
4 WWS_EN	Wait State Enable for writes This bit controls the wait state to be added for all write transactions. The state of this field effects full width writes as well as narrow width writes. This field is used to enable the memories to run at half the system frequency using punch-through clock 0 No extra wait state for Write operations 1 One extra wait state for Write operations
3 MEMACC_WAIT	Wait State Enable for ECC for RMW operations This bit controls the wait state required for ECC calculation on RMW path. The state of this field has no effect on the response latency of reads. 0 No extra wait state for RMW operations. 1 One extra wait state for RMW operations.
2 RWS_EN	Wait State Enable for high memory access time This bit controls the wait state to be added to the best case RAM array access time for read. The state of this field effects read as well as partial write operations. 0 RAM read takes 1 cycle. 1 RAM read takes 2 cycles.
1 FIXED_BURST_EN	Fixed Burst Enable This bit controls burst transaction behavior for a FIXED type burst. 0 Fixed burst not supported. PRAMC signals SLVERR. 1 Fixed burst is supported by PRAMC.

15.4 Functional description

15.4.1 Read/write

The RAM controller generates chip-select, write enable, array address, write mask and write data as inputs to the RAM array. The RAM controller captures read data from the RAM array and drives it onto the AXI4 System bus. To facilitate operation at higher frequencies, the RAM controller can optionally incur a single wait state on the AXI system bus.

Table 176. AXI read-write channels stall conditions

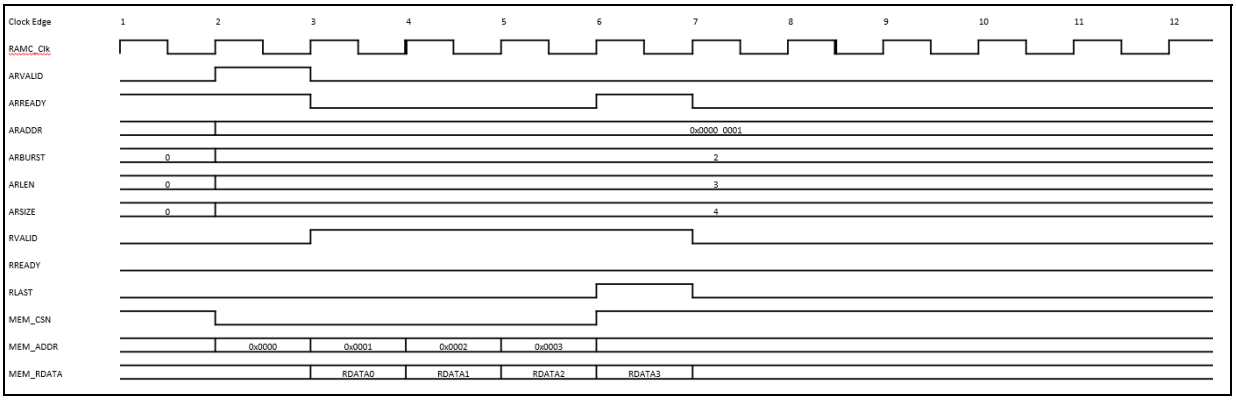
Channel	Condition
Write address channel	Write address channel is stalled under following conditions: – Waiting for data for 1st write transfer – Write address received while ongoing read – Write address received at same time as read It is stalled until the last write to the current transaction is initiated to the memory.
Write data channel	Write data channel is stalled under following conditions: – Waiting for Write Address to start the write transfer – Write data received while ongoing read – Write data received at same time as read – Every write transfer with access size < ECC granularity It is stalled until write transaction can make progress after read has completed.
Write response channel	Write response channel is stalled under the following condition: – Master is not ready to accept the write response.
Read address channel	Read address channel is stalled until last read for the current transaction is initiated to the memory. (NUM_OUTSTANDING_READS =1)
Read data/response channel	Read data channel is stalled until the Master is ready to accept the read response.

15.4.1.1 Reads

Read transfers of any size can be configured to complete with either a zero or one additional wait state response on the AXI system bus.

The read proceeds in a similar way even if a write request arrives at the same time as the read request.

Figure 106. Read with CR[RWS_EN]=0



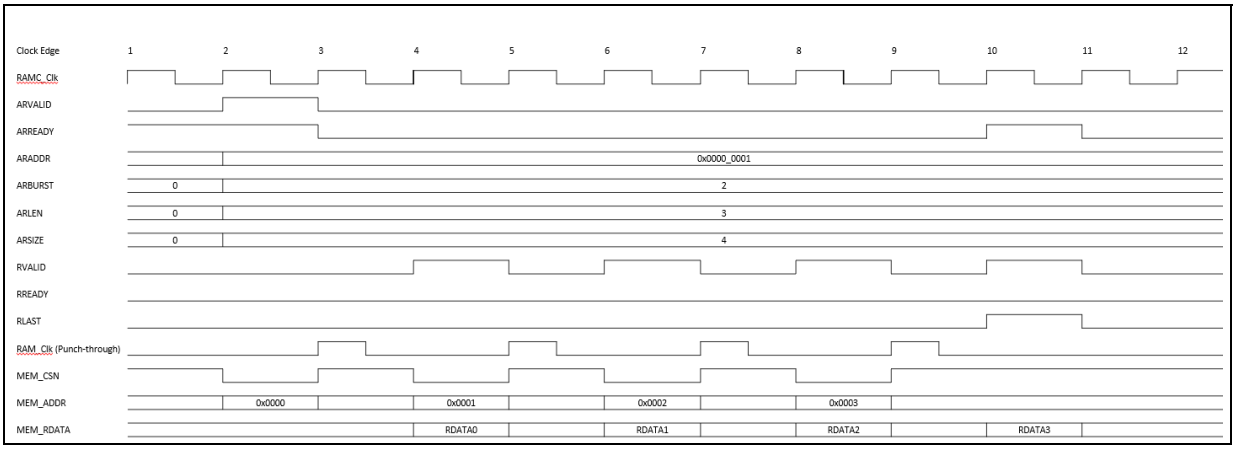
15.4.1.2 Optional read wait state

The RAM controller can be optionally programmed for use when operating with slower memory access times.

A random initial access takes two clock cycles to complete if CR[RWS_EN]=0 and a WRAP4 burst have an access time of 2-1-1-1.

A random initial access takes three clock cycles to complete if CR[RWS_EN]=1 and a WRAP4 burst have an access time of 3-2-2-2.

Figure 107. Read with CR[RWS_EN]=1

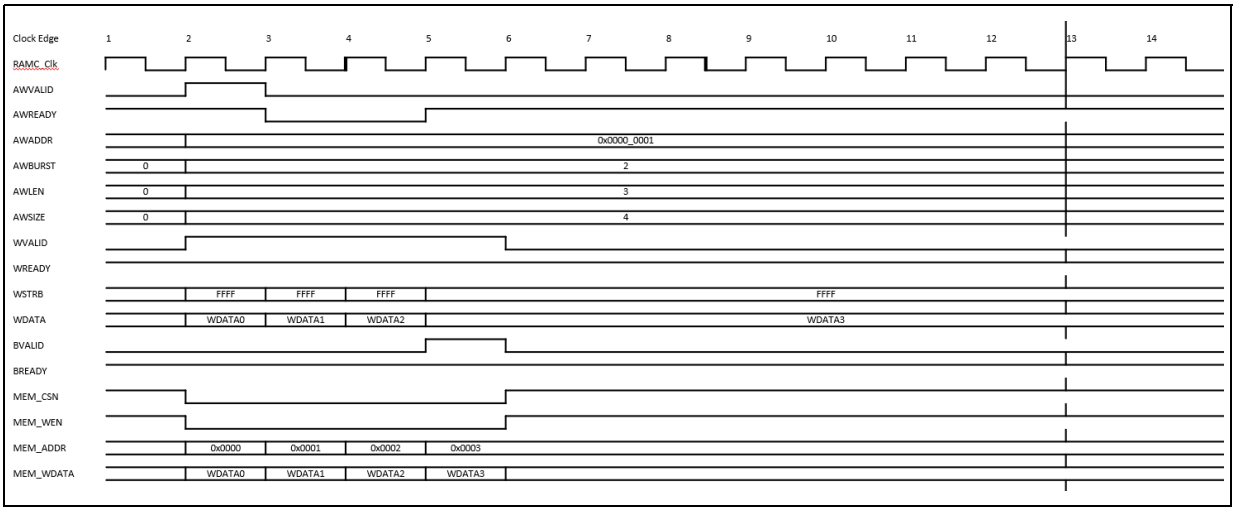


15.4.1.3 Writes

15.4.1.3.1 256/218/64-bit writes

Full width writes (256/128/64 bits) are completed in single cycle or in two cycles to maintain pulse width requirements for memories. The RAM controller can be optionally programmed for this requirement.

Figure 108. Full width write with CR[WWS_EN]=0



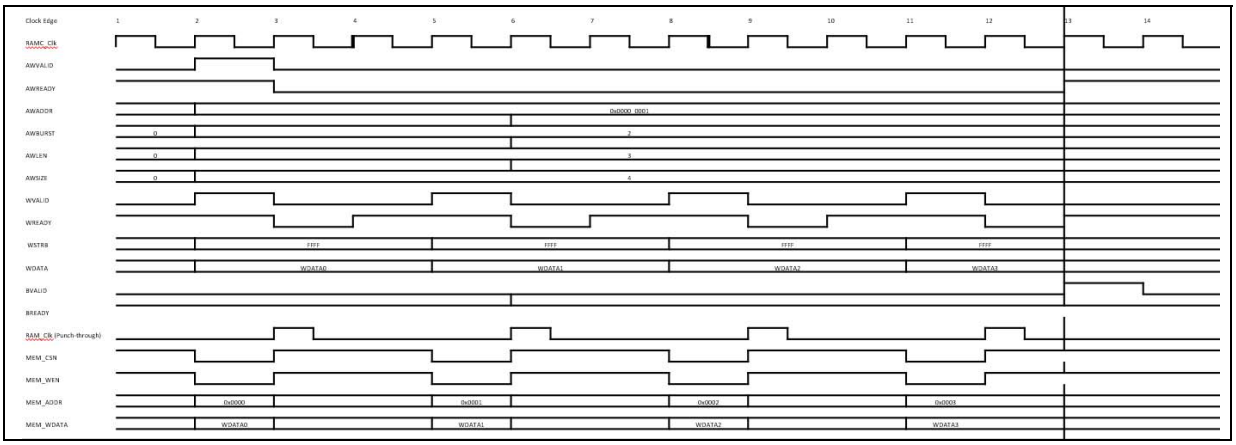
15.4.1.3.2 Optional Write Wait State

The RAM controller can be optionally programmed for use to meet pulse width requirements at higher frequencies.

A write takes one clock cycle to complete if CR[WWS_EN]=0 and a WRAP4 burst complete in 1-1-1-1.

A write takes two clock cycles to complete if CR[WWS_EN]=1 and a WRAP4 burst complete in 2-2-2-2.

Figure 109. Full width write with CR[WWS_EN]=1



15.4.1.3.3 Less than 64 bits writes

Writes of size less than 64 bits require a read-modify-write operation as a consequence of the ECC coding scheme's 64 bits granularity. In case of a read-modify-write operation, PRAM controller performs SEC/DED on the read data. The write data is merged into the appropriate byte lanes along with potentially corrected read data. A new codeword is generated based on the newly formed data. The newly formed data and its associated checkbits are subsequently written to the RAM.

Valid wait state combinations for RMW operations are:
CR[RWS_EN], CR[WWS_EN], CR[MEMACC_WAIT]=000
or
CR[RWS_EN], CR[WWS_EN], CR[MEMACC_WAIT]=111

Figure 110. Narrow width write with CR[RWS_EN]=0; CR[WWS_EN]=0; CR[MEMACC_WAIT]=0

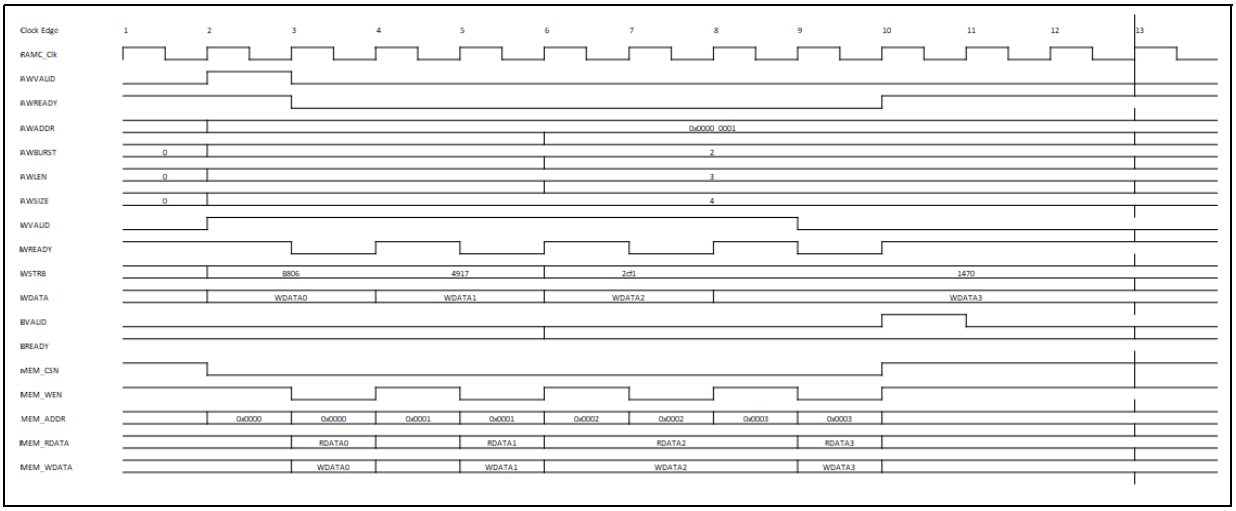
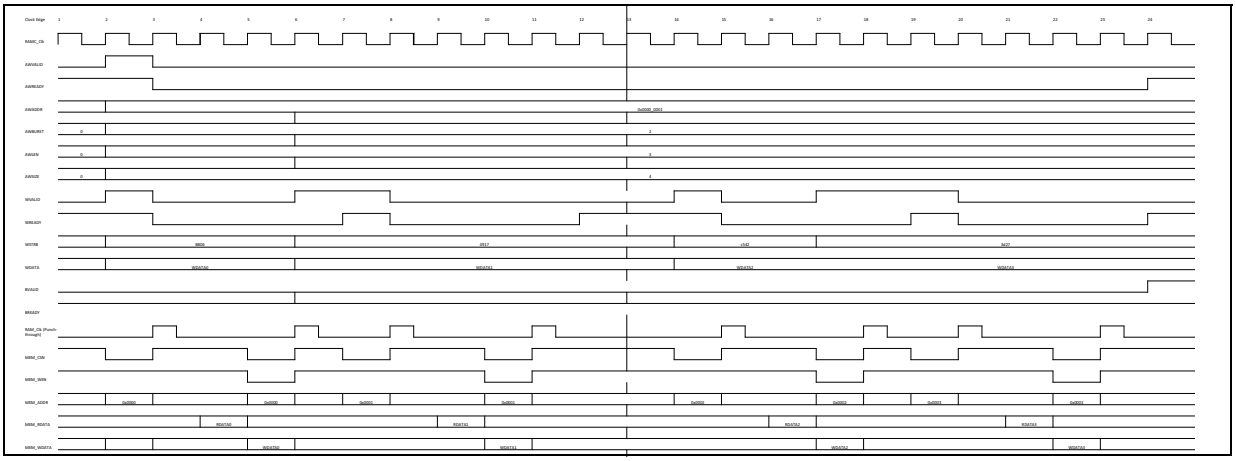


Figure 111. Narrow width write with CR[RWS_EN]=1; CR[WWS_EN]=1; CR[MEMACC_WAIT]=1



15.5 Initialization/application information

It is essential to initialize each memory address to a known value before reading to avoid generation of spurious ECC errors. Without writing an address to a known value first, a read from this address most likely generates uncorrectable ECC events.

15.6 Safety considerations

15.6.1 Data Protection

The RAM controller implements single error correction double error detection using 64-bit ECC scheme used in Cortex®-M7.

15.6.1.1 Write

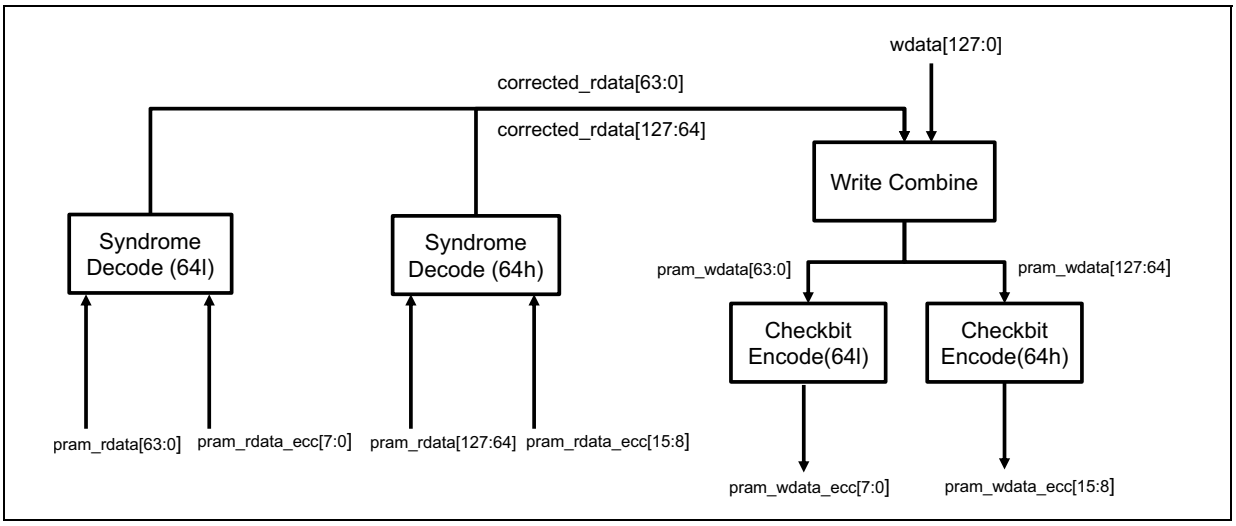
The RAM controller receives data along with checkbits from the NIC.

In case of a full-width write, there is no ECC check and the data is written along with the checkbits in the RAM memory.

In case of a partial write operation, the RAM controller performs a read-modify-write operation. The RAM controller reads the data stored in the target location and performs SECDED on the read data. It then merges it with the data received from master and generates the new checkbits to be written to the RAM memory. Any single or multi-bit error detected during the read from memory is reported to the MEMU2. It responds with an error response on the system bus in case of a multi-bit error.

The datapath for a RMW operation is highlighted below.

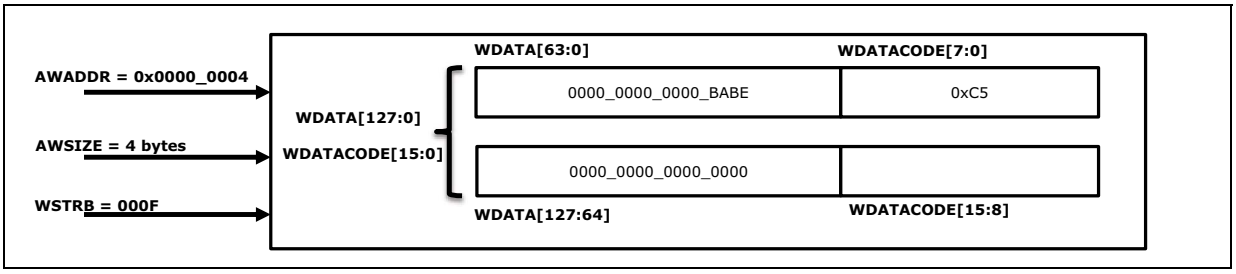
Figure 112. PRAM Controller ECC management for read-modify-writes



The RAM Controller uses the following checkbit manipulation technique to ensure faults in the PRAM control logic associated with performing the read-modify-write operation are ultimately detected by e2e ECC:

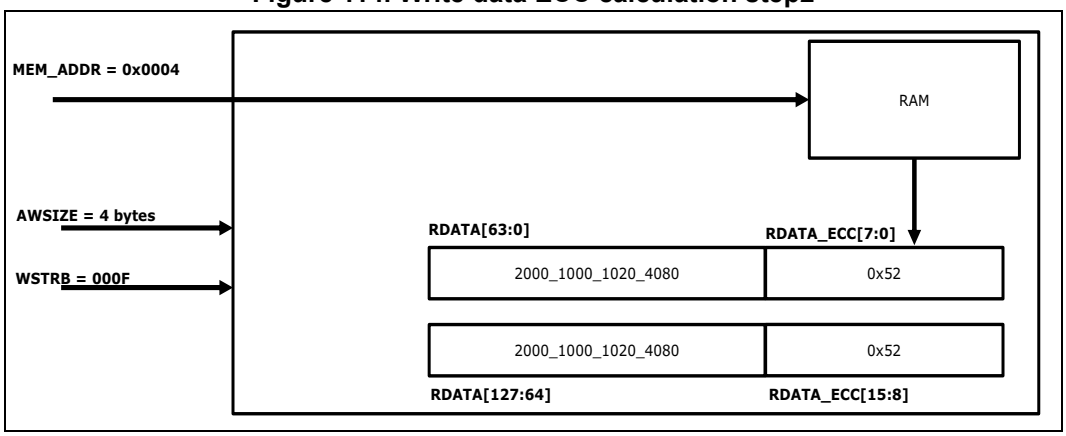
- On a partial write request, the write data is presented on AXI system bus in correct byte lanes with non-pertinent byte lanes zeroed out. The associated check-bit is based on this “zero-padded” write data.
  For example, the case where Master performs a 16-bit write at address 0x0000_0004.

Figure 113. Write data ECC calculation step1



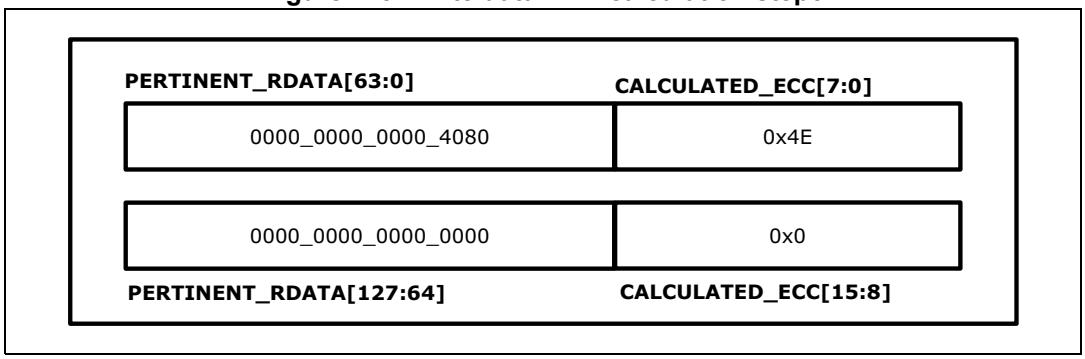
- RAM controller treats that request as a read-modify-write operation and initiates a read to that address. An ECC check and potential single-bit correction is performed on the data returned from the RAM. If a non-correctable ECC event is detected, an error is signaled on the AXI system bus. Additionally, the ECC event is reported to MEMU2.

Figure 114. Write data ECC calculation step2



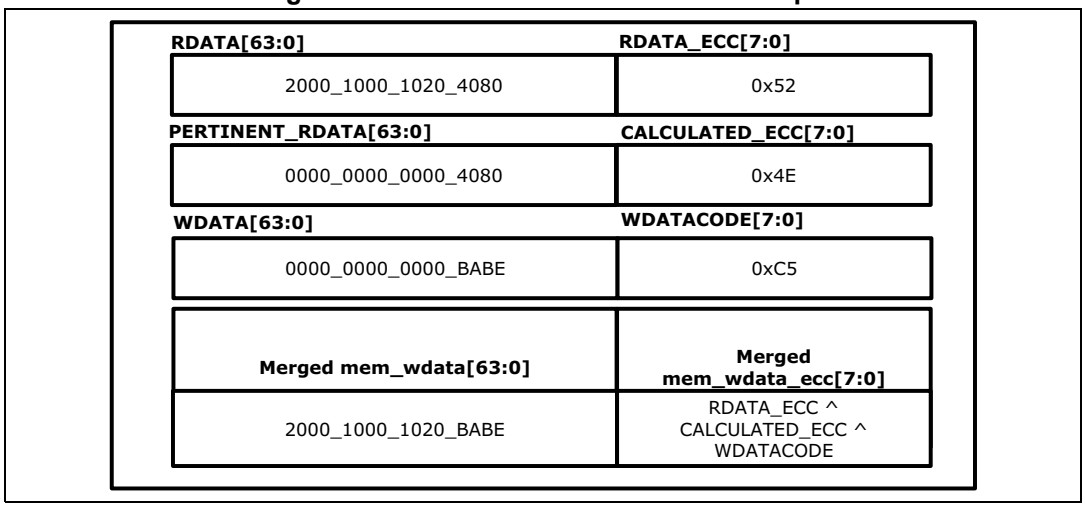
- RAM Controller isolates the pertinent byte lanes of the read data by zero-padding the non-pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.

Figure 115. Write data ECC calculation step3



- For WR data generation: merge the non-pertinent read data bytes with the write data in appropriate byte lanes.
  For WR ECC generation: take the checkbit value stored in RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with 64-bit data to be written to the array.
  This is done in two chunks of 64 bits for 128 bits of data.

Figure 116. Write data ECC calculation step4



This method calculates the checkbit value on a read-modify-write transaction through a series of data manipulations, thus taking care to avoid discarding of original checkbits provided by the requesting master.

Malfunction of ECC logic described above may result in corruption of error event reporting. Thus, EDC after ECC check is performed on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to FCCU.

15.6.1.2 Read

The RAM controller performs an EDC check on the data read from the RAM. Any single/multi-bit error does not gate the read response path while it is reported to the MEMU2.

15.6.2 MEMU2 interface

There is one channel to the MEMU2 for each 64-bit chunk of data. The same channel is used to report an error on a read or an RMW operation to MEMU2.

15.6.2.1 Memory Update in case of ECC error

Memory is not updated if an uncorrectable error is detected for a narrow write transfer. The following table lists the possible scenarios.

Table 177. Memory updates in case of ECC error

Error in 4th chunk (bits [255:192])	Error in 3rd chunk (bits [191:128])	Error in 2nd chunk (bits [127:64])	Error in 1st chunk (bits [63:0])	Memory updated
Correctable Error	Correctable Error	Correctable Error	Correctable Error	Yes
Uncorrectable Error	X	X	X	No
X	Uncorrectable Error	X	X	No
X	X	Uncorrectable Error	X	No
X	X	X	Uncorrectable Error	No

15.6.3 Memory feedback monitor

The feedback checker provides functional safety coverage between the PRAM controller and the RAM array.

The function of the monitor relies on a feedback path between the PRAM controller and the RAM array. The RAM provides latched address and control feedback outputs as indication of received inputs when a RAM access is initiated. The feedback monitor uses this feedback information to verify the expected transaction. A mismatch indicates a failure in the transmission path between the PRAM controller and the RAM array. The event is forwarded to the fault collection and control unit.

15.7 Exclusive monitor

The local monitor enables the RAMs to be used for semaphores between processes running on different masters. The local monitor exclusive granule size is 8 bytes.

A load exclusive transaction on the bus marks a 64-bit memory region as exclusive. A store exclusive passes the local exclusive monitor check if it accesses the same 64-bit memory region and returns the exclusive monitor to open access state. If a store exclusive accesses an address outside the marked 64-bit region, then the store exclusive instruction fails the exclusive monitor check and the monitor is returned to open access state.

There are multiple exclusive monitors to enable it for multiple masters. The number of exclusive monitors is controlled by NUM_PROCESSORS.

15.7.1 Support for Exclusive monitor behavior

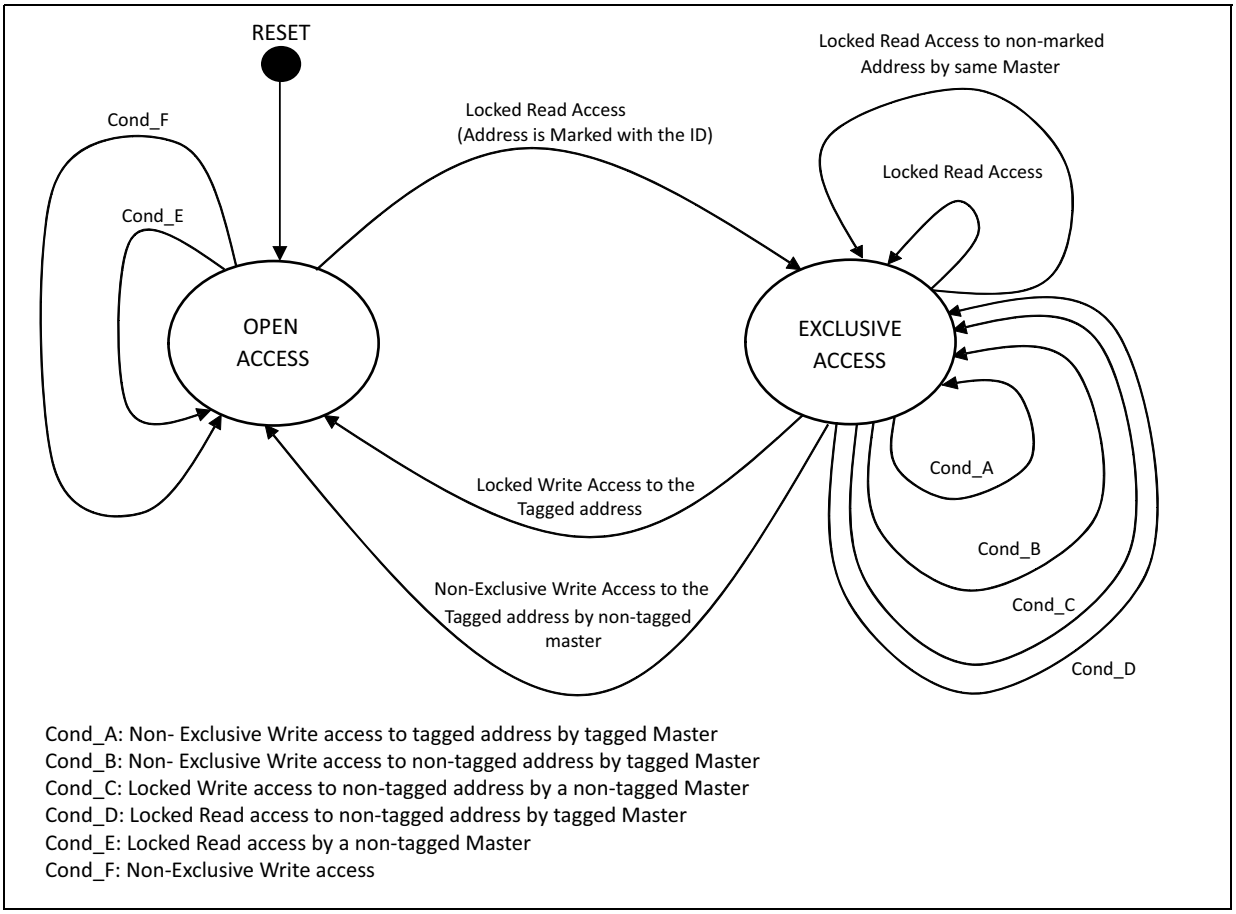
15.7.1.1 Features

- Support exclusive requests aligned to local monitor concept as specified in Arm Core
- Exclusive reservation granularity would be 64 bytes
- Support one exclusive monitor per processing element
- Aligned with the implementation as defined in Arm architecture reference manual ARMv8_A for Exclusive monitor

15.7.1.2 Functional description

The exclusive monitor supports two states - OPEN and EXCLUSIVE. After Power on Reset, the state machine remains in OPEN state. Based on the type of operation it transitions to EXCLUSIVE state. The details of the state machine are shown below.

Figure 117. Exclusive monitor state machine



Some specific points are:

- Any non-exclusive STORE do not generate a no-write condition.
- WRITE operation is not performed when STREX is not hit in any monitor in the following cases:
  - Store exclusive to non-tagged address by tagged master when the monitor is in OPEN state.
  - Store exclusive to non-tagged address by a non-tagged master.

16 Nonvolatile memory platform controller (NVMPC)

16.1 Introduction

16.1.1 Overview

This document describes the features of the embedded non-volatile memory (NVM) platform controller (NVMPC). The block has these functions:

- It acts as an interface between the system AXI4 master and the NVM program/erase controller.
- It converts the protocols between the system bus and the dedicated NVM array interface.

The block consists of two modules, implementing an AXI4 read interface and an AXI4 write interface. The modules can be instantiated independently.

16.1.2 Features

- System interface:
  - Support for one 64-bit AXI4 target bus interface for write interface block
  - Support for two 64-bit AXI4 target bus interface for read interface block
  - Support for full burst size/length capability, including support of narrow transfers for read interface block
  - Support for full burst size/length capability, but narrow transfers for write interface block are not supported
  - Support for all response type for read channel and write channel
  - Support for separate IPS interface for register programming
- NVM interface:
  - Supports configurable size of NVM memory with configurable NVM address bits of array addresses available
  - Support of programmable access timing (wait-state) allowing use in a wide range of frequency targets
  - Support of NVM access pipelining (ACP) to improve performance and access latency reduction
  - Support for NVM array interface of 256-bit read data bus + 64-bit write data bus
- Functional safety:
  - Support for protection of address and control signal of AXI4 interface using an e2e ECC
  - Support for fault detection in address re-encoding check from feedback address from NVM
  - Support of end-to-end ECC (e2eECC) coverage with the read and write data, the corresponding ECC code words traverse the entire data path
  - Support for reporting of single, double and multi-bit NVM ECC events on a 64-bit double word boundary
  - Support for data valid check with the internal counter and input data valid signal from NVM

- Security:
  - Support of protection for NVM areas (regions) using PASS_REGIONS set for NVM read access
  - Support of protection for NVM areas (regions) using a sideband signal of size pALTSEL_REG, the secure-exclusive concept to protect secure dedicated regions from non-secure core read accesses and to execute write accesses from the NVM alternate interface
  - Support of interrupt flags for the accesses with data collision feature for both non-secure and secure cores
- Selectable features:
  - Support for Configurable (default is 2) Outstanding Address for Read Interface Block
  - Support of OTA-X1 feature

The NVMPC module supports the following modes of operation:

- Write transfers, implemented by the AXI4 write interface block
- Read transfers, implemented by the AXI4 read interface block

Such modes of operation are described in detail in *Section 16.8*.

The read interface supports single 64-bit AXI4 slave port for Global, Read address and Read data channels. It has full burst size/length capability, including support of narrow transfers, and operates as a bridge from the protocol to the custom NVM read protocol (described in *Section 16.8.1.2*); the NVM wait state number is configurable and access to pipelining is supported, with maximum address phase anticipation of one clock cycle (described in *Section 16.6*).

One outstanding address is accepted.

The write interface translates 64-bit write accesses from AXI4 master to 64/32bit accesses to the NVM interface; narrow transfers are not supported by this module and the NVM write is expected to be performed in a single clock cycle, with no capability of stalling the interface (described in *Section 16.6*). Single address is accepted from the master during a burst.

Along with the read and write data, the corresponding ECC codewords traverse the entire datapath to support end-to-end ECC (e2eECC) coverage (described in *Section 16.9*).

16.2 Block interface

The block interface is composed by the following protocol interfaces:

- *Memory map and register definitions*
- *AXI4 global/read data/read address interface*
- *Nonvolatile-memory read interface*
- NVMPC interrupt enable

The AXI4 write interface block is composed of the following protocol interfaces:

- *AXI4 write data/write address/write response interface*
- *Nonvolatile-memory write interface*

Global safety features are managed by specific sideband signals, described in *Section 16.9*.

16.3 Memory map and register definitions

16.3.1 Memory map

In the AXI4 read interface an IPS programming model mapped to a standard 16 KB on-platform peripheral slot is implemented. The NVMPC allows access to the programming model by all system bus masters.

Table 178. NVM controller memory map

Address offset	Register	Location
0x0000	NVM PC configuration register 1 (PFCR1)	*Section 16.3.2.1*
0x0004–0x0007	Reserved	
0x0008	NVM PC configuration register 3 (PFCR3)	*Section 16.3.2.2*
0x000C	NVM PC Status register (PFSR)	*Section 16.3.2.3*
0x0010	NVM PC fault latching enable register (FLTENA)	*Section 16.3.2.4*
0x0014	NVM PC fault forcing register (FLTFRC)	*Section 16.3.2.5*
0x0018	NVM PC fault status and clear register (FLTSCR)	*Section 16.3.2.6*
0x001C	NVM PC fault address register (FLTADR)	*Section 16.3.2.7*
0x0020–0x033F	Reserved(1)(2)	

1. Transfer error is not issued at reserved address (such as 0x0300)
2. Additional registers exist to handle the data memory collision avoidance functionality. Refer to the SR5E1x security reference manual.

16.3.2 NVM controller registers

16.3.2.1 NVM PC configuration register 1 (PFCR1)

The configuration register 1 (PFCR1) is used to configure the read operations, interconnected to the NVM.

Offset 0x0000 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	Reserved(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APC				RWSC				0	0	0	0	0	0	Reserved(1)	
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0

1. Read/write to this field does not return any error.

Figure 118. Platform NVM configuration register 1 (PFCR1)

Table 179. PFCR1 field descriptions

Field	Description
15:13 APC	Address Pipeline Control This field controls the number of cycles that a subsequent NVM read from the opposite AXI port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1. 000 Pipelined access to the NVM disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 Reserved 011 Reserved 1xx Pipelined access to the NVM is disabled and one wait state is inserted before a subsequent access can be initiated.
12:8 RWSC	Read Wait State Control This field controls the number of wait-states to be added to the best-case NVM array access time for reads. The best-case NVM array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the AXI4 read interface and the actual read access time of the NVM. The required settings are documented in the device data sheet. Higher operating frequencies require non-zero settings for this field for proper NVM operation, as computed by the formula: $N_{RWSC} = floor\ (T_{actime} / T_{aclk})$ **Note:** Do not change the NVM RWSC while fetching from the NVM itself.

16.3.2.2 NVM PC configuration register 3 (PFCR3)

The configuration register 3 (PFCR3) is used to configure the protection, interconnected to the NVM.

Offset 0x0008 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BAF_DIS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 119. Platform NVM configuration register 3 (PFCR3)

Table 180. PFCR3 field descriptions

Field	Description
15 BAF_DIS	BAF_DIS This bit is used to prevent the re-execution of the BAF code. **Note:** This bit can be written only once and it is cleared after reset.

16.3.2.3 NVM PC status register (PFSR)

The Status Register (PFSR) is used to Sample the status information from NVMC and NVM.

Offset 0x000C Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NVMPCFLHBUSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 120. NVM PC Status register (PFSR)

Table 181. PFSR field descriptions

Field	Description
0 NVMPCFLHBUSY	NVMPCFLHBUSY This bit reflects the flh busy pin value connected to NVMPC instance.

16.3.2.4 NVM PC fault latching enable register (FLTENA)

This NVM PC fault enable register (FLTENA) is used to enable or disable the error forwarding to MEMU or FCCU.

When setting this register, the corresponding bit in the register enables the fault forwarding.

When the bit of the register is cleared, the fault is not forwarded to FCCU or MEMU, however the FLTSCR latches the fault internally anyway.

Offset 0x0010 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	NVMCTED	NVMCDBC	NVMCSBC	NVMDTED	NVMDDBC	NVMDSBC	NVMENCE	NVMCEDC	NVMDEDC	NVMPCENC	NVMPCENSWAP
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	1	1	0	1	1

Figure 121. Platform NVM fault latching enable register (FLTENA)

Table 182. FLTENA field descriptions

Field	Description
10 NVMCTED	NVM Code triple ECC error
9 NVMCDBC	NVM Code double ECC error
8 NVMCSBC	NVM Code single ECC error
7 NVMDTED	NVM Data triple ECC error **Note:** DATA fault forwarding is not implemented.
6 NVMDDBC	NVM Data double ECC error **Note:** DATA fault forwarding is not implemented.
5 NVMDSBC	NVM Data single ECC error **Note:** DATA fault forwarding is not implemented.
4 NVMENCE	NVM Address encoding error
3 NVMCEDC	NVM Code EDC error
2 NVMDEDC	NVM Data EDC error
1 NVMPCENC	NVMPC Address encoding error
0 NVMPCENSWAP	NVMPC SWAP Address error

16.3.2.5 NVM PC fault forcing register (FLTFRC)

This NVM PC fault forcing register (FLTFRC) is used to force the error latching in order to check the error reporting path. When setting this register, the corresponding bit in the FLTSCR is set and a signal to the fault collector is asserted. The FLTENA masks the error forced with FLTFRC.

Offset 0x0014 Access: Read/Write(1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	NVMCTED	NVMCDBC	NVMCSBC	NVMDTED	NVMDDBC	NVMDSBC	NVMENCE	NVMCEDC	NVMDEDC	NVMPCENC	NVMPCENSWAP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reading back to this register reads 0.

Figure 122. Platform NVM fault forcing register (FLTFRC)

Table 183. FLTFRC field descriptions

Field	Description
10 NVMCTED	NVM Code triple ECC error
9 NVMCDBC	NVM Code double ECC error
8 NVMCSBC	NVM Code single ECC error
7 NVMDTED	NVM Data triple ECC error
6 NVMDDBC	NVM Data double ECC error
5 NVMDSBC	NVM Data single ECC error
4 NVMENCE	NVM Address encoding error
3 NVMCEDC	NVM Code EDC error
2 NVMDEDC	NVM Data EDC error

Table 183. FLTFRC field descriptions (continued)

Field	Description
1 NVMPCENC	NVMPC Address encoding error
0 NVMPCENSWAP	NVMPC SWAP Address error

16.3.2.6 NVM PC fault status and clear register (FLTSCR)

This NVM PC fault status and clear forcing register (FLTSCR) is used to latch the fault and error coming from NVM or generated internally by NVMPC. The bit is set when the corresponding fault is detected or fake fault is injected using FLTFRC register. This latching is not gated by FLTENA register. The bit can be cleared by write 1 to a bit.

Offset 0x0018 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	NVMCTED	NVMCDBC	NVMCSBC	NVMDTED	NVMDDBC	NVMDSBC	NVMENCE	NVMCEDC	NVMDEDC	NVMPCENC	NVMPCENSWAP
W						w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 123. Platform NVM fault status and clear register (FLTSCR)

Table 184. FLTSCR field descriptions

Field	Description
10 NVMCTED	NVM Code triple ECC error
9 NVMCDBC	NVM Code double ECC error
8 NVMCSBC	NVM Code single ECC error
7 NVMDTED	NVM Data triple ECC error
6 NVMDDBC	NVM Data double ECC error

Table 184. FLTSCR field descriptions (continued)

Field	Description
5 NVMDSBC	NVM Data single ECC error
4 NVMENCE	NVM Address encoding error
3 NVMCEDC	NVM Code EDC error
2 NVMDEDC	NVM Data EDC error
1 NVMPCENC	NVMPC Address encoding error
0 NVMPCENSWAP	NVMPC SWAP Address error

16.3.2.7 NVM PC fault address register (FLTADR)

This NVM PC fault address register (FLTADR) is used to latch the start address of transaction where there is fault detected from NVM. Note that this register captures only the first address of the transaction while the MEMU is notified by the exact address where the Error is happened in case of multiple errors are detected in a transaction.

Note: Write to this register does not produce a transfer error.

Offset 0x001C Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FADR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FADR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 124. Platform NVM fault address register (FLTADR)

Table 185. FLTADR field descriptions

Field	Description
31:0 FADR	Fault Address

16.4 AXI4 global/read data/read address interface

On an incoming AXI4 burst request, that is when both ARREADY and ARVALID are *high1*, the master request address/controls are latched. During the following clock cycle the read FSM starts converting the master request into multiple memory read requests; on each transaction, 64-bit of data are returned to the master on the RDATA[63:0] bus along with transaction qualifiers.

In order to decouple the memory read flow from the AXI4 master access beats, a FIFO buffer is implemented in each read interface FSM block. Valid data is notified to the AXI4 master by asserting the RVALID interface signal if FIFO is not empty; the master then acknowledges data by asserting the RREADY signal, and the FIFO top is sent to the master.

16.4.1 Operation

16.4.1.1 Read cycles / FIFO hit

A dedicated read FIFO is filled by the data flow from the memory and the burst is stalled only if the FIFO is full, maximizing the gain given by the access pipelining. On a memory read, the line is stored in the FIFO. The RVALID signal is *high1* when the FIFO is not empty or when the data is ready from the memory array.

16.4.1.2 Read cycles / hit

Single cycle read responses to the AXI4 are possible with the AXI4 read interface when the requested read access was previously loaded into one of the page buffers. In these “buffer hit” cases, read data is returned on the system bus with a zero wait-state response.

16.4.1.3 Read cycles / miss

In the event of a buffer miss, a NVM access is initiated.

On bursts requests issued by the interface *allocated master*, incoming read data from the NVM array is stored in the least-recently updated 256-bit page read buffer of the selected way; concurrently the critical double-word of NVM read data is forwarded onto the system AXI4 bus.

If the memory access was the direct result of a bus transaction, the corresponding page buffer is updated and marked as most-recently-used.

If the memory access was the result of a speculative prefetch to the next sequential line, it is stored into the least-recently-used buffer.

16.5 Nonvolatile-memory read interface

The AXI4 read interface read block is tightly interfaced to the embedded non-volatile-memory and implements the memory custom protocol.

16.5.1 Operation

A memory read cycle is composed by a single cycle address phase, followed by a data phase of one or multiple clock cycles. The number of wait states is configurable via the

PFCR1.RWSC register, and it is forwarded to the memory by the ADWS (cfg_adws) interface bus. Read accesses are terminated under control of the appropriate wait state settings. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temperature).

The memory data/errors is registered on the last cycle of the data phase and made available to the user on the next cycle, while the memory asserts the Data Ready (p0_data_rdy) signal. A data error is signaled in case of uncorrectable ECC errors by asserting Triple Error(p0_triple_error). A physical memory address decoding error is signaled by asserting the Address Encoding Error (p0_aee_error).

When an error response is received, the AXI4 read interface interface returns a slave code on the RRESP port to the master. A SLVERR code on the RRESP signal is then returned to the master.

Read cycles from the memory array are initiated by driving a valid access address on Read Address (p0_addr[23:0]) and asserting Read Strobe Signal (p0_rd_en) for the required setup (and hold) time before (and after) the rising edge of AXI clock.

The AXI4 read interface then waits for the assertion of the Data Ready from the memory before sampling the read data on Response Data Bus (p0_rdata[255:0]).

Any error (ECCx, EEE, AEE) signaled by the NVM for a data word, is considered effective at the time when the AXI4 master consumes the data: during that clock cycle the NVMPC signals any error, which has been stored in FIFO along with the data.

Data affected by error at source and that never used read by the AXI4 master never triggers an error by the NVMPC block.

16.5.1.1 Read accesses pipelining

Back-back memory accesses can take advantage of the read access pipelining (APC) feature, which is implemented to improve the data throughput. Refer to *Section 16.8.1.2.1*.

16.5.1.2 Read timing coherency

Parallel to the NVM read counter, a read counter is implemented in the AXI4 read interface.

During a read access, the read block counters are used to check the correct assertion of the data valid signal from the memory, and in case of mismatch an error is reported to the AXI4 master.

If the Data Ready (p0_data_rdy) is not asserted when the internal count ends, the error is flagged and the SLVERR code is returned by the RRESP bus.

16.5.1.3 Read stall mechanism

The memory protocol supports a stall-while-operation feature.

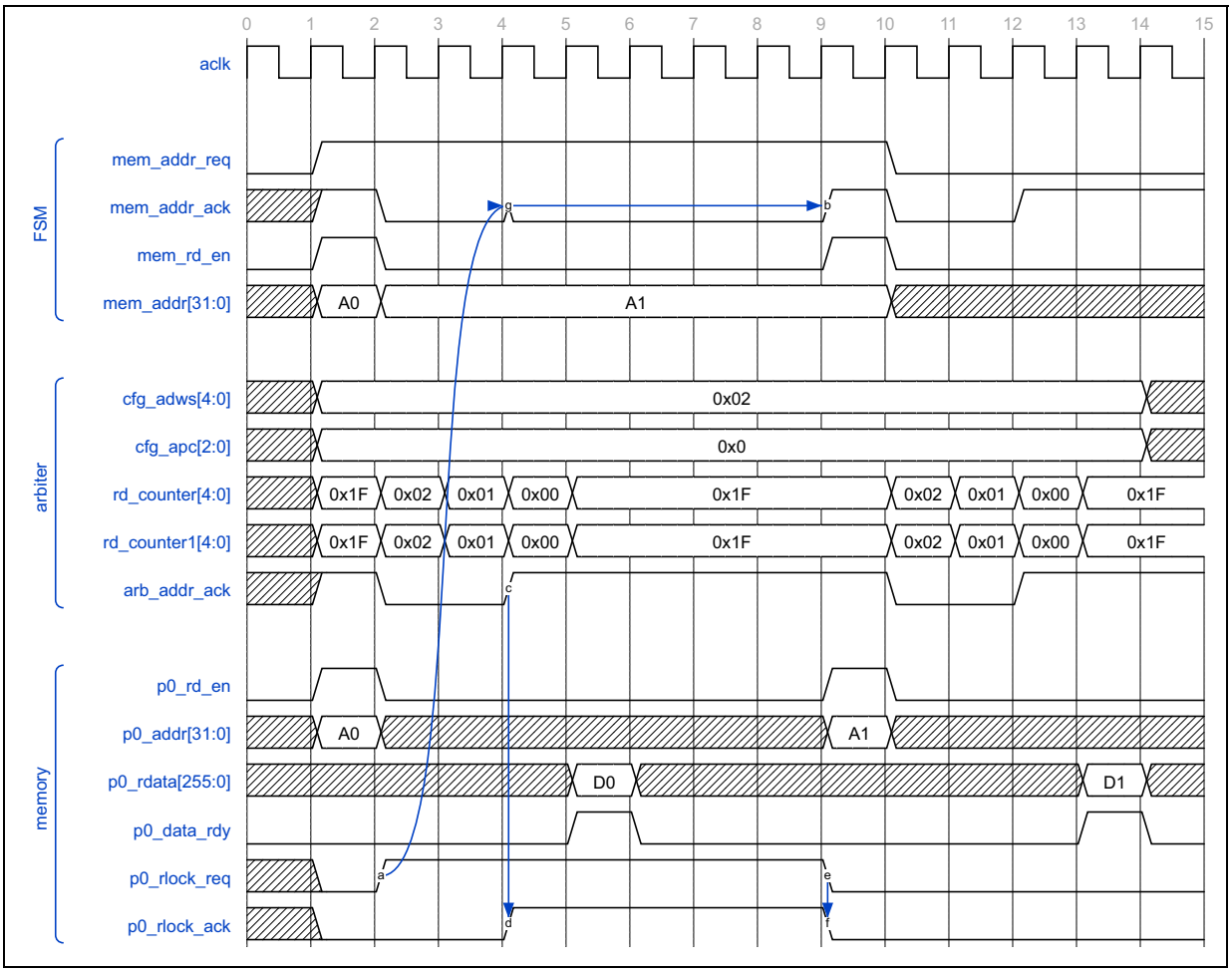
If required, the memory can operate back-pressure and suspend any read burst in order to perform a higher priority task. It is in charge of the memory to initiate and release the stall.

The feature is implemented by a handshake, initiated by the assertion of the Read Lock Request (p0_rlock_req) signal from memory; this signal inhibits any further read to be issued, while the ongoing access completes. The handshake is completed by the assertion of the Read Lock Acknowledge (p0_rlock_ack) to the memory. The release of the Read Lock Request (p0_rlock_req) exits the stall-while-write condition.

In the same clock cycle when the Read Lock Acknowledge (p0_rlock_ack) is asserted, the Read Penalty (p0_rd_penalty_en) signal is sampled. This allows the NVM memory to request a read penalty to be added to the programmed RWSC count, in a safe manner that is without affecting any already initiated read access. If penalty is enabled, the NVMPC starts adding a latency based the assertion time of Read Penalty (p0_rd_penaly_en) signal to the ADWS (cfg_adws) information to the NVM.

Penalty is disabled by performing a new handshake, clearing the Read Penalty (p0_rd_penalty_en) signal.

Figure 125. NVM stall mechanism



16.6 AXI4 write data/write address/write response interface

The AXI4 write interface IP implements single AXI4 write interface, for write buffer filling of the target NVM.

On an incoming AXI4 burst request that is when both AWREADY and AWVALID are *high1*, the master request address/controls are latched. The following clock cycle, the write FSM starts converting the master request into multiple memory write requests; on each transaction, 64 bits of data are provided by the master on the WDATA[63:0]; the master asserts WVALID to *high1* until data is acknowledged by the slave which asserts WREADY to

high1. The WSTRB[7:0] signals are used to flag the valid bytes in the data word, one bit per byte.

16.6.1 Operation

Each AXI4 master write transaction is translated into two successive NVM accesses; the memory is accessed with no wait states, and no idle cycle is inserted between the write transactions by the AXI4 write interface.

16.6.1.1 Write cycles

Write cycles to the memory array are initiated by driving a valid access address on Read Address (p0_addr[23:0]) and asserting Write strobe (p0_wr_en). For write cycles data is provided on Write Data (p0_wdata[63:0]) and Write Data ECC (p0_wdata_ecc[7:0]). The size of the write data is indicated by Write 64-bit Strobe; the Write 64-bit Strobe (p0_wr64) signal is *high1* if the WSTRB[7:0] indicates that valid data is present in the low order and high order 32-bit words in the current Write Data (p0_wdata[63:0] to memory.

Again, the AXI4 write interface drives the address and control information for the required setup time before the rising edge of AXI Clock (axi_clk), and continues to drive address and control information on the memory interface bus for the required amount of hold time beyond the rising edge of AXI Clock (axi_clk).

16.7 Nonvolatile-memory write interface

16.7.1 Operation

A single write access to the NVM_P0 interface is always a single cycle operation with no stall capabilities. The address/data phase is initiated when the Write Strobe (p0_wr_en) signal is *high1*, and in the same clock cycle the Address (p0_waddr[*23*:0]), Write Data (p0_wdata[63:0]), Write Data Parity (p0_wdata_ecc[7:0]) and Write 64-bits Strobe (p0_wr64) must be valid.

The NVM write buffer is filled by a sequence of data writes, and it can be accomplished by transferring data in 64-bit or 32-bit words.

When the Write 64-bit Strobe (p0_wr64) signal is *high1*, the full Write Data (p0_wdata[63:0]) is accepted by the NVM. When the Write 64-bit Strobe (p0_wr64) signal is *low0*, only 32 bits of the full Write Data p0_wdata[63:0]) is accepted by the NVM:

- The Write Data (p0_wdata[31:0)] if Address (p0_waddr[2]) is *low0*
- The Write Data (p0_wdata[63:32]) if Address (p0_waddr[2]) is *high1*

The Write Data Parity (p0_wdata_ecc[7:0]) is always checked against the full Write Data (p0_wdata[63:0]), regardless of the Write 64-bit Strobe (p0_wr64) signal level.

16.8 Functional description

The AXI4 read interface block architecture is composed by the following functional sub-block:

- *AXI4 read slave controller*

The AXI4 write interface block interface is composed by the following functional sub-block:

- *AXI4 write slave controller*

Global safety features are managed by specific sideband signals, described in *Section 16.9*.

16.8.1 AXI4 read slave controller

16.8.1.1 FIFO read buffer operation

The internal FIFO is a three lines by 256-bit buffer; the buffer is capable of stalling the read FSM on its quasi-full condition, asserted when two lines are filled; on this event, the remaining line holds the data from the already started memory read.

On any address update by the FSM, the data is searched in the FIFO. In case of FIFO hit, the memory read is skipped and the address is stepped to the following beat with zero latency. The FIFO hit condition is computed through combination from the current address and the most recent memory read address.

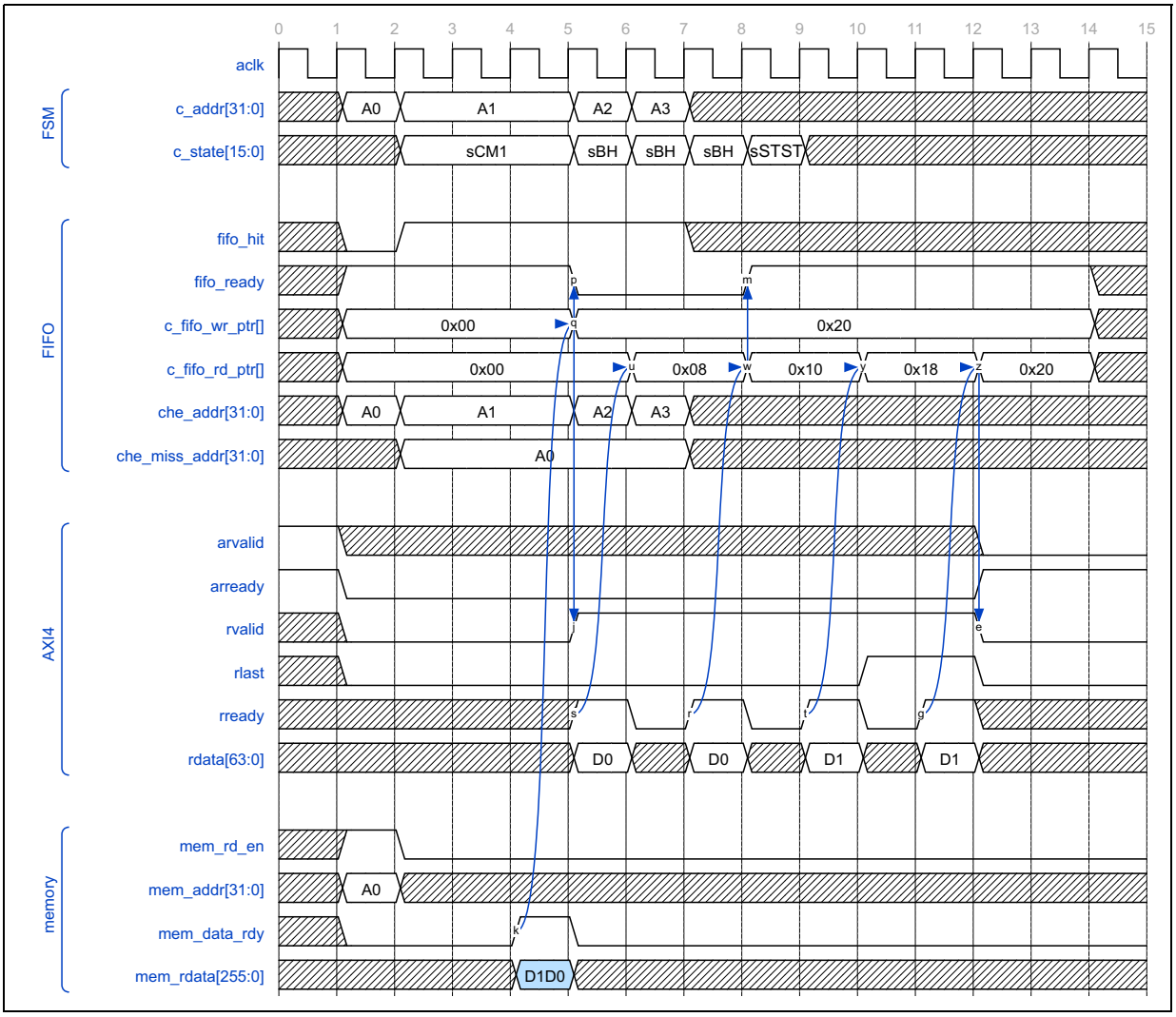
The FIFO is filled by any memory read, and the write pointer is incremented by the actual memory block data size (0x10 or 0x20 bytes).

The FIFO block is in charge of asserting the AXI4 RVALID signal if any valid data is in the buffer. The assertion of the RREADY signal by the AXI4 master increments the FIFO read pointer by the actual ARSIZE[2:0] value. When the read pointer equals the write pointer the RVALID signal is de-asserted.

When the master asserts RREADY to *high1*, if any FIFO line is valid, the older FIFO line is returned and discarded, else the memory data line is directly routed to the RDATA[63:0] bus as soon as it is ready.

In case of incrementing accesses to a code memory area the full Read Data (p0_rdata[255:0]) is valid; since this information is valid over four successive AXI4 transfers, the 192 MSb of data is returned from the read FIFO avoiding a new read requests and minimizing the memory bandwidth usage.

Figure 126. FIFO read buffer operation



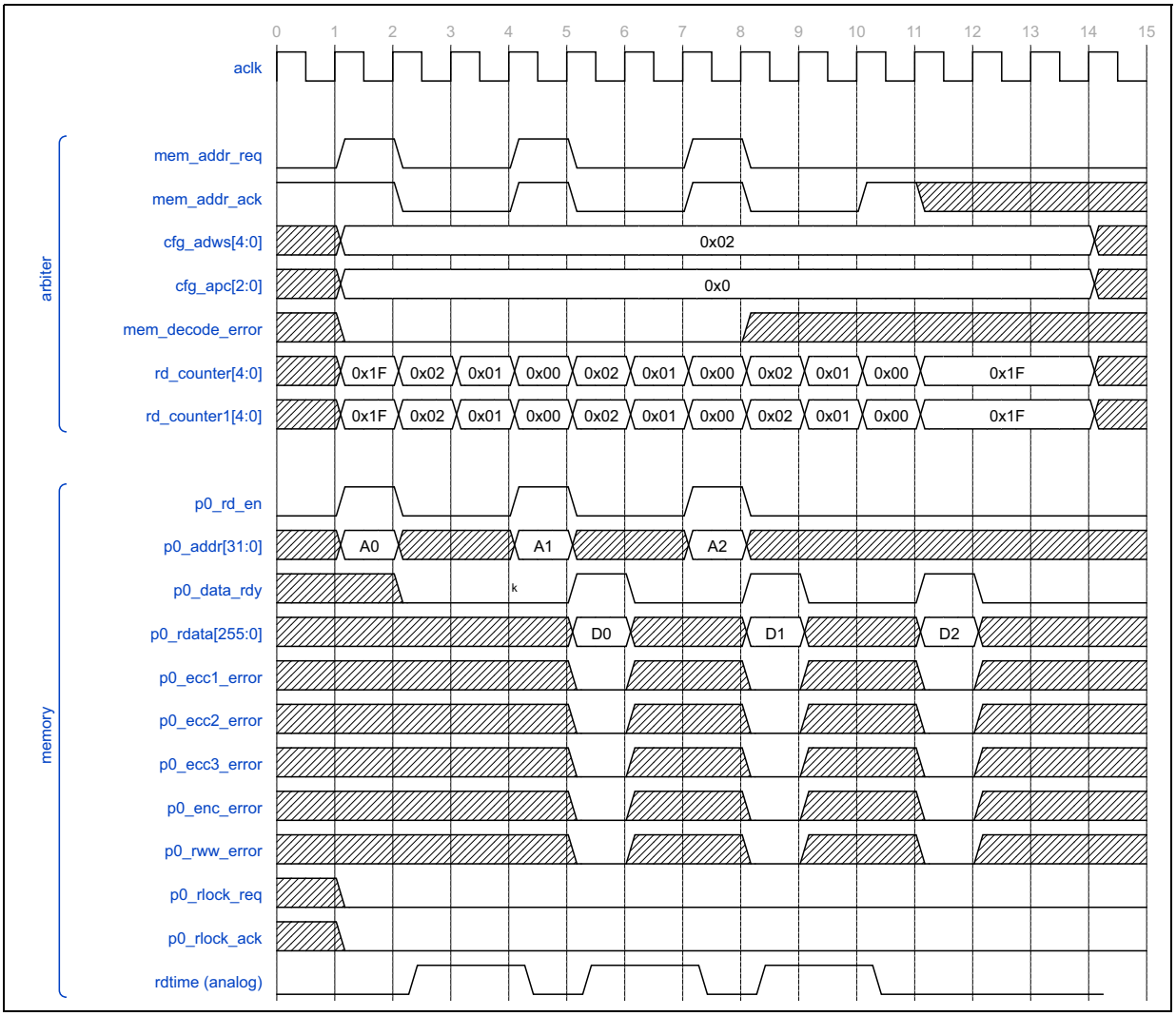
16.8.1.2 Memory access arbiter

An arbiter is in charge to serialize the arriving access requests.

The memory acknowledge signal takes into account the actual memory resource availability, evaluating the internal Address Acknowledge (s_mem_addr_ack) signal as follows (refer to *Section 16.8.1.2.1*):

- If APC is enabled (PFCR1.APC=0x1), memory is available if the Read Counter (rd_counter) is {0x01,0x00,0x1F}.
- If APC is disabled and additional wait state is required (PFCR1.APC=0x4), memory is available if the Read Counter (rd_counter) is {0x1F}.
- If APC is disabled and no additional wait state is required (PFCR1.APC=0x0), memory is available if the Read Counter (rd_counter) is {0x00,0x1F}.

Figure 127. Memory read with pipeline disabled



16.8.1.2.1 Read access pipelining

Accesses to the NVM array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete.

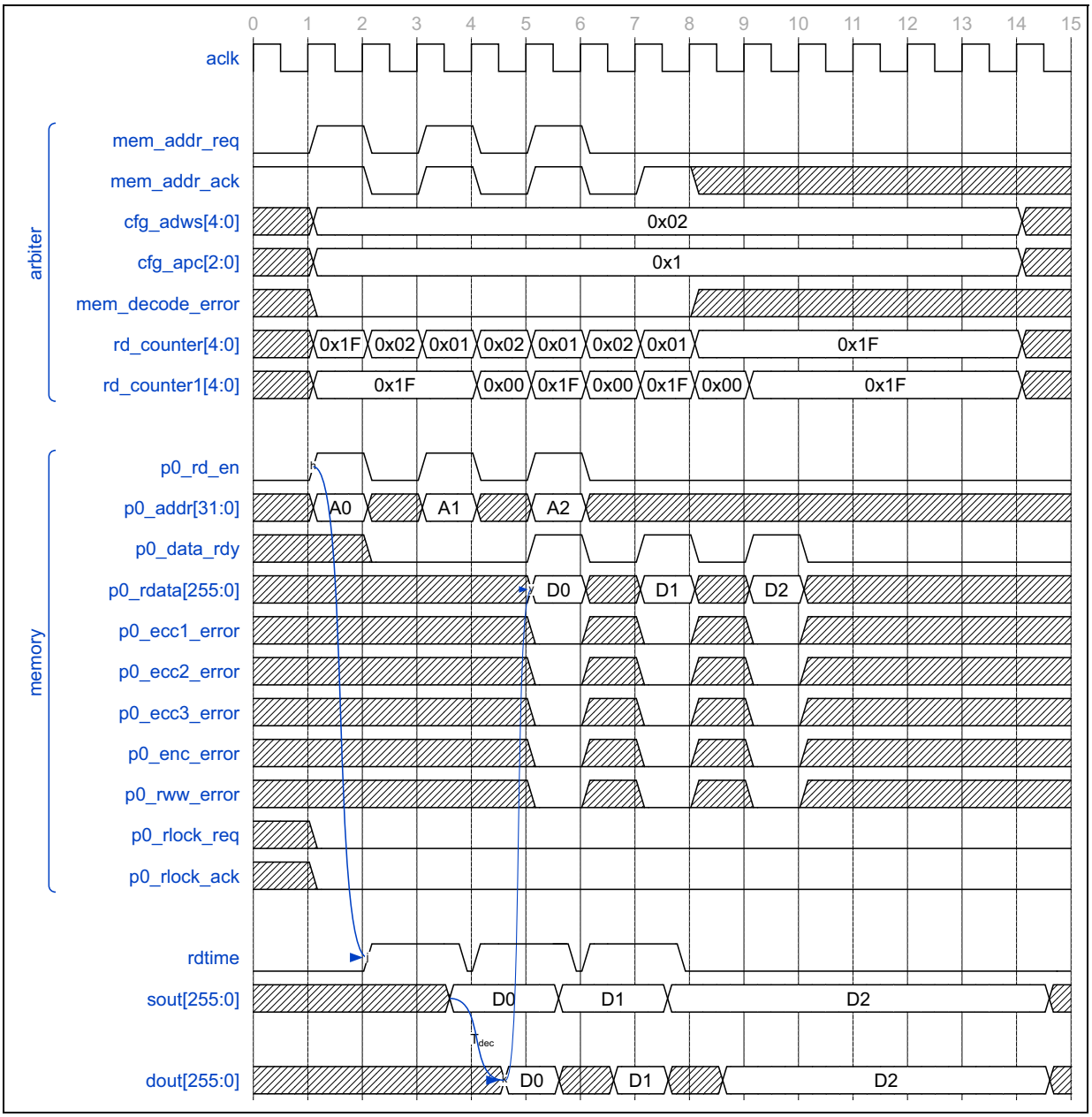
NVM access pipelining (ACP) allows for improved performance by reducing the access latency seen on back-back read cycles targeting the NVM array. This is made possible because the NVM hard macro internally latches the data after the asynchronous analog access time T_{rdtime}, and a new read strobe is allowed after this delay; data then passes through the NVM soft macro multiplexers and ECC decoders, having a combinational delay of T_{dec}, and get registered before the Read Data (p0_rdata) pins. The full access time is such that $T_{rdtime}+T_{dec}<(RWSC+1)T_{clk}-T_{clk,setup}$. The access pipelining feature triggers a new read just after T_{rdtime} leveraging the fact that the NVM hard macro internal latches actually implement a pipeline on data.

Any hit to the read FIFO interrupts the pipelining flow, inserting a non-pipelined read after the hit.

Address pipelining is enabled by setting the PFCR1.APC register field.

Not all combinations of APC (Address Pipeline Control) and RWSC (Read Wait State Control) settings are recommended or supported. Refer to the device data sheet for guidance. The integrator is strongly encouraged to verify these settings based on actual silicon characterization.

Figure 128. Memory read with pipeline enabled



16.8.1.2.2 AXI4 read interface memory map

The read interface decoding scheme is controlled by a parameter, which selects the NVM port number. Implemented port numbers are 0 to 5; depending on the physical memory port to be controlled; each instance of the read NVMPC implements a subset of the full read address map.

Any transfer issuing an unmapped address returns dummy data (all bits being ‘0’), along with a DECERR AXI4 slave response.

The decoder blocks take in input the system address, and return the memory local address and the Decode Error (mem_decoder_error) signal, asserted if the input address does not lie inside the controller address space.

For security purposes (refer to *Section 16.10*), the block has the input cfg_region_lock[19:0], and operates a comparison of that signal to the region currently accessed, based on a static hard-coded decoding table. In case of accesses to a protected region, the Protect Error (mem_protection_error) signal is asserted to the requesting block.

16.8.1.2.3 AXI4 write interface memory map

The write interface implements the full decoding table.

Different AXI4 address maps are allowed for write and read; address ranges mapped to the ‘Kite NVM' read-only interface can require different mapping for write.

16.8.2 AXI4 write slave controller

16.8.2.1 FIFO write buffer operation

The write buffer is a single 128-bit word storage. On an AXI4 write beat, the data is stored into the buffer, and transferred to the NVM_P0.

16.9 Safety

16.9.1 End2End ECC

Dedicated logic blocks are implemented to protect the data and control path from the AXI4 interface and towards it.

16.9.1.1 NVM data E2E ECC

The AXI4 read interface does not alter the data and parity from the NVM_P0. It only blindly routes the information from the memory to the AXI4 master.

The NVM must implement the same H-matrix as the AXI4 master.

The NVM uses DECTED and it includes a logic to transform from Kite SECDED to DECTED and use internally, while before giving back data to NVMC, the NVM transforms from TEDSEC to Kite SECDED.

No ECC encoder/decoder is implemented by the NVMPC block for the end-to-end data path. Safety relies on the matching of the NVM and Arm core ECC polynomials. The NVMPC only routes the ECC parity word along with the data.

16.9.2 Error management

The NVMPC block collects and routes data/address errors which could be internally detected or signaled by the NVM block.

The FLTSCR/FLTADR registers report the error status, and give the capability to clear the error status. Any error is latched in the FLTSCR and signaled by a level signal on the NVMPC interface.

Error signaling can be selectively enabled and disabled by the FLTENA register.
Fake errors can be forced by the FLTFRC register.

16.9.2.1 NVMPC address re-encoding error

On every read request to the NVM_P0, the pre-decoding SoC address is compared to the address computed by applying a dedicated inverse encoding function to the Address (p0_addr) output.

If these two values do not match an error is reported (*high1* active signal) and the error is latched in the FLTSCR. An OKAY response is sent to the AXI4 master. The actual data given by NVM is returned to the master, along with OKAY response.

16.9.2.2 NVM data parity errors

The NVMPC collects the error information related to a data word returned by the NVM.

Data parity errors (TED/DEC/SEC) are stored in the FIFO and triggered when the master actually consumes the data.

SLVERR is generated for uncorrectable ECC Error found in Code region. The Data Region uncorrectable Error does not translate to SLVERR. The FLTEN only enables/disables the reporting to MEMU. For uncorrectable data Error, there is fixed Error response data({0xE7F000F0_0xE7F000F0_0xE7F000F0_0xE7F000F0}) and Response data code of 0x4747 is returned.

The NVMPC, based on its embedded address decoding table, generates dedicated signals for data/code errors starting from the collective signal returned by the NVM.

- Single Bit Parity Error (p0_ecc1_error): memu_ecc11_single_error (CODE) and memu_ecc10_single_error (DATA).
- Double Bit Parity Error (p0_ecc2_error): memu_ecc11_double_error (CODE) and memu_ecc10_double_error (DATA).
- Triple Bit Parity Error (p0_ecc3_error): memu_ecc11_multi_error (CODE) and memu_ecc10_multi_error (DATA).

The FLT* registers have dedicated bits for NVM data/code parity errors.

16.9.2.3 NVM Read-While-Write error

The NVMPC collects the error information related to a data word returned by the NVM.

Read-While-Write Error (RWE) is stored in the FIFO and triggered when the master actually consumes the data.

The error is managed by returning a SLVERR to the AXI4 master and the returned data from NVM is send back to master.

16.9.2.4 NVM address re-encoding

The NVMPC collects the error information related to a data word returned by the NVM.

Address Encoding Error (AEE) is triggered as soon as it is signaled by the NVM.

The FLT* registers have dedicated bits for NVM AEE error.

In case of address encoding error, the Master is given back OKAY response with the data from NVM.

16.9.2.4.1 ECC after ECC error

The NVMPC collects the error information related to a data word returned by the NVM. ECC after ECC Error (EDC) is triggered as soon as it is signaled by the NVM.

All FLT registers have dedicated bits for NVM EDC error. In case of ECC after ECC error, the Master is given back OKAY response with the data from NVM.

16.9.2.4.2 OTA-X1 SWAP Error

The NVMPC detects the OTA-X1 swapping using two swap signals along with ota_enable. If there is ota_enabled asserted while the swap bits are not equal, the NVMPC generates the OTA-X1 SWAP Error to FCCU. There is no specific response to Master for this Error.

The FLT* registers have dedicated bits for OTA-X1 SWAP Error.

16.10 Security

16.10.1 Read NVM access protection

A selective protection on a predefined set of NVM areas (regions) is implemented in the NVMPC.

The NVM blocks allocation to each of the PASS_REGIONS region is hard-coded and embedded in the NVMPC design.

During any NVM read, the region number is looked-up, and it is compared to the information sent by the cfg_region_lock[31:0] interface bus signal.

If the region is protected (cfg_region_lock bit line is *high1*) the read is not done to the NVM and ALL0 data is returned to the master, along with SLVERR response.

16.10.2 Data flash management

RWW partition 3 of NVM memory array 0 (as shown in *Figure 129*) hosts sectors for both Host EEPROM emulation (Host Data Flash) and HSM EEPROM emulation (HSM Secure Data Flash).

This setup can lead to potential critical system behavior (RWW exceptions) in the following cases:

- Read request on HSM Secure Data Flash while Host Data Flash is under modifications
- Read request on Host Data Flash while HSM Data Secure Flash is under modifications

In case, Host or HSM application requires to access (read) its own data, while a sector of the same partition 3 is under modification by the other master, the SW must be written taking care of avoiding contentions (RWW exceptions) and Host and HSM application SW cannot be developed in an independent way.

An enhanced HW feature, called “HSM vs Host Data Collision Avoidance”, is available in order to manage the above access contentions in a transparent way for SW.

For details about the way to enable and use this new HW feature, refer to the security reference manual.

16.11 Over the air (OTA-X1) support

16.11.1 Overview

The OTA feature can be enabled by the customer through a bit of Global System Configuration DCF.

The OTA feature is implemented for a portion of the non-HSM Code NVM sectors only, no HSM and NVM Data sectors can be updated via OTA. Refer to the NVM memory map in the Memory Map chapter for details on the organization of the NVM sectors. The following table shows the partitions and OTA contexts that can be swapped when OTA feature is enabled.

Table 186. OTA context mapping

Context	RWW
A	1, 2
B	2, 1

The image programming can be done by any core. It is possible to program a new SW image into the NVM while the application is still running. Once activated, the new SW image is visible at the same addresses as the old one.

16.11.2 Architecture

The product NVM can be organized in two different ways, depending on bit OTA_EN of the Global System Configuration DCF. When the OTA functionality is disabled, the NVM addresses are fixed in HW: no dynamic swap of any NVM region is possible.

When the OTA functionality is enabled, the NVM is divided into 3 regions:

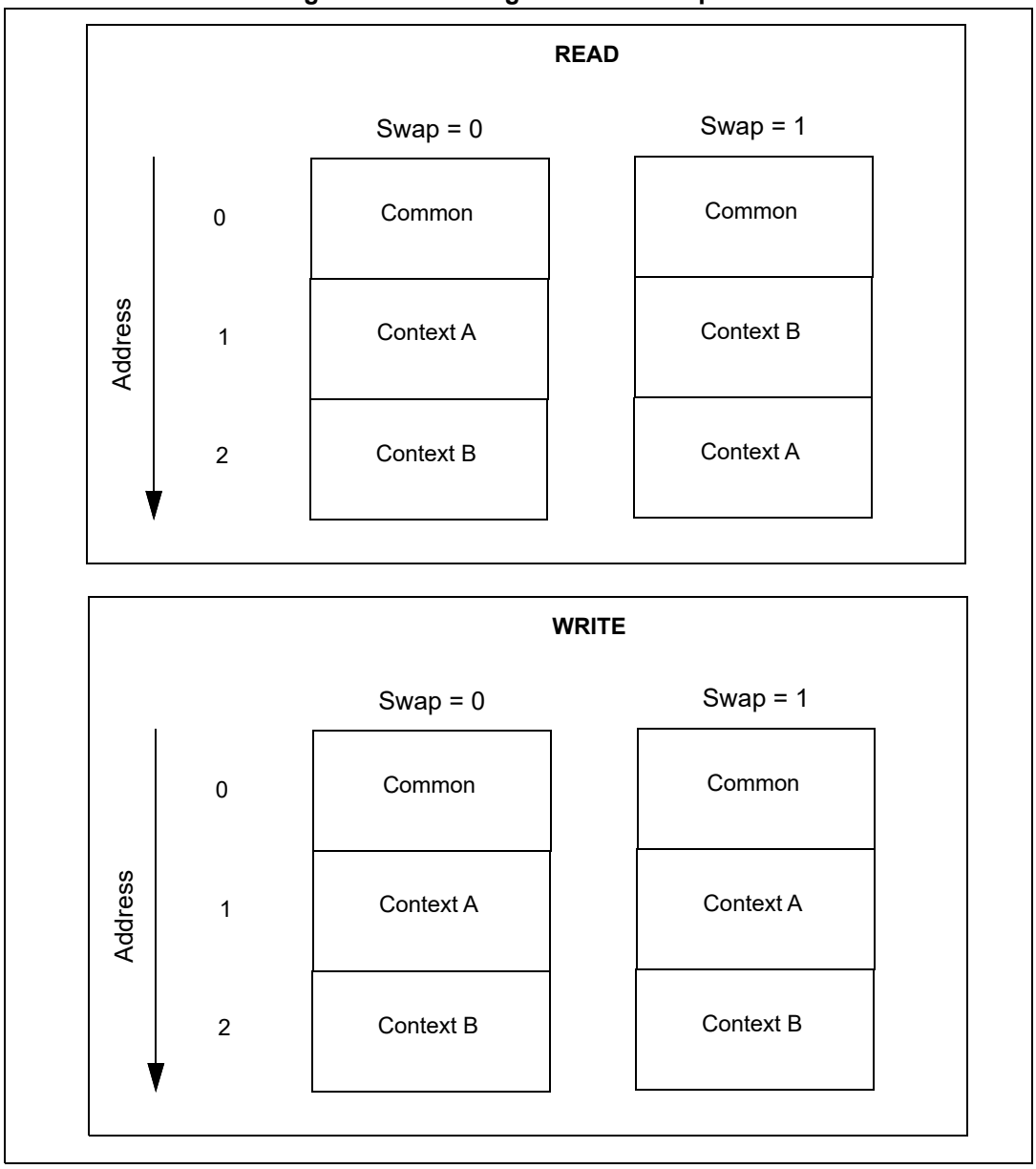
- A common region, which can be updated only in the garage (that is using some specific tools).
- Two OTA contexts, which can be swapped to enable one or the other software images.

The three regions are seen by the MCU masters at different address spaces, as shown in *Figure 129*.

Note that whereas the OTA contexts swap for a read operation (that is, the active context is always seen at address “1”), they do not swap for a write operation, that is, the non-active context, where the new image is going to be programmed, is seen at address “1” or “2, depending on which is the active context.

The reason for this is that the NVM identifies the sector during erase with a combination of bits in a register and write address: because of this it is not possible to erase a sector belonging to context A passing an address within region “2”.

Figure 129. OTA regions address space



16.11.2.1 Functionality

The OTA feature is split in 3 phases:

1. Programming the new image
2. Programming the piece of information that identifies which is the active context
3. Enabling the swap

The following sections describe each of these phases in detail.

16.11.2.1.1 New image programming

Note: The section refers to the case when an update of an existing SW is done. The case of the very first SW programming at factory is detailed later on.

Starting from the situation of an existing SW already programmed in the active OTA context, the new image is always programmed in the no-active context.

The SW doing this operation knows in which address region is the non-active context programmed.

In order to pass this piece of information to the SW, a register is implemented in the SSCM: the SSCM reports into this register the logical address of the non-active context, so that the SW can use it to program the new image.

16.11.2.1.2 OTA context swap programming

The piece of information that characterizes which is the active context is written in a UTEST NVM sector.

The information is written in sequential order, in slots of 16 bytes starting from the address 0x1FF8_3800.

The space allocated to store the signature of the active context inside UTEST is 2 Kbyte, allowing to have 127 updates. The data programmed in each slot is irrelevant and can be user-defined.

It is mandatory that the last slot of the sector is never programmed.

16.11.2.1.3 OTA context swap enabling

After the new OTA image and the related signature have been programmed, the new image becomes the active one, where the SW executes the code from. The process of enabling of the new active OTA context is done only upon a destructive reset. During this phase, as explained later on, the OTA signature is looked up and the piece of information indicates which OTA context is active and is latched.

A long functional reset does not change the OTA context, which remains fixed to the value latched during the last destructive reset. Nevertheless, upon a functional reset the checks of the signature integrity are still executed.

A short functional reset has no relevance from OTA point of view: nothing is read from the NVM<var> and the latched outcome (that is which OTA is active) is preserved.

The IP dedicated to decode the OTA signature is the SSCM, which is active during the reset phase.

The SSCM does the following operations:

1. Evaluate whether the OTA feature is enabled or not (as reported by a bit of the Global System Configuration DCF)
2. If the OTA feature is enabled:
   a) Read UTEST space reserved for signature storage and determine which context is the active one.
   b) Drive the signals to the SOC indicating which OTA context is the active one.
3. If the OTA feature is not enabled:
   a) Drive the signals to the SOC indicating the context A as the active one.

16.11.2.1.4 Signature reading and decoding

The signature slot can contain any user-defined data.

From the SSCM point of view, it is only important to detect that something has been programmed in a signature slot, in order to distinguish between programmed or not-programmed.

The “goal” of the SCCM is to determine which is the active context and pass this piece of information to the SOC.

The information is sent to the SOC in the form of two sideband signals, which are generated by two set of triple-voted flip-flops, called TVF1 and TVF2.

If the OTA is disabled, no signature is read and the SSCM defaults to the Context A as the active context.

If OTA is enabled, following a destructive reset, the SSCM linearly scans the signature portion within the UTEST sector till it finds the latest programmed slot. Note that it is mandatory that the last slot of the sector is always left non-programmed. A counter, which starts from 0, increases for every slot recognized as programmed. Once the last programmed slot is found, the SSCM checks the counter value: if it is an even number, then context A is selected; if it is an odd number, then context B is selected. If the counter is 0, it means that no OTA has been done yet and the NVM still contains the original code programmed by the customers during their production phase. In this case, the context A is activated. In addition, the SSCM saves:

- The address of the last programmed signature slot in *Section 19.2.1.5: OTA signature address location register (OSAL)*.
- The identifier of the non active partition in *Section 19.2.1.6: Non active context logical address register (NACLA)*.

17 Embedded flash memory

17.1 Introduction

The flash memory module serves as electrically programmable and erasable nonvolatile memory, for instructions and data storage.

The flash memory module is a nonvolatile, solid-state silicon memory device consisting of blocks of single transistor storage elements, an electrical means for selectively adding (programming) and removing (erasing) charge from these elements, and a means of selectively sensing (reading) the charge stored in these elements.

The flash memory module has two functional units:

- The flash core: it is composed of a common high voltage HV block and a flash array which can contain one or more flash partitions. Each flash partition is composed of arrayed nonvolatile storage elements, sense amplifiers, row decoders and column decoders. The arrayed storage elements in the flash partitions are subdivided into physically separate units referred to as blocks.
- The memory interface: it contains the registers and logic which control the operation of the flash core and contains two main ports: the Array Interface and the Registers Interface. The memory interface is also the interface between the flash memory module and a Bus Interface Unit (BIU) and contains the ECC and redundancy logic.

17.1.1 Overview

The 2208 KB flash memory module contains four array partitions. Read-While-Write operations are only possible for separate partitions, meaning fetching/reading from one (or more) partition(s) while a program/erase operation is active on another partition. Each module counts as an RWW partition.

EEPROM emulation is managed as 4 × 16 KB blocks in one partitions, referred to as Data flash.

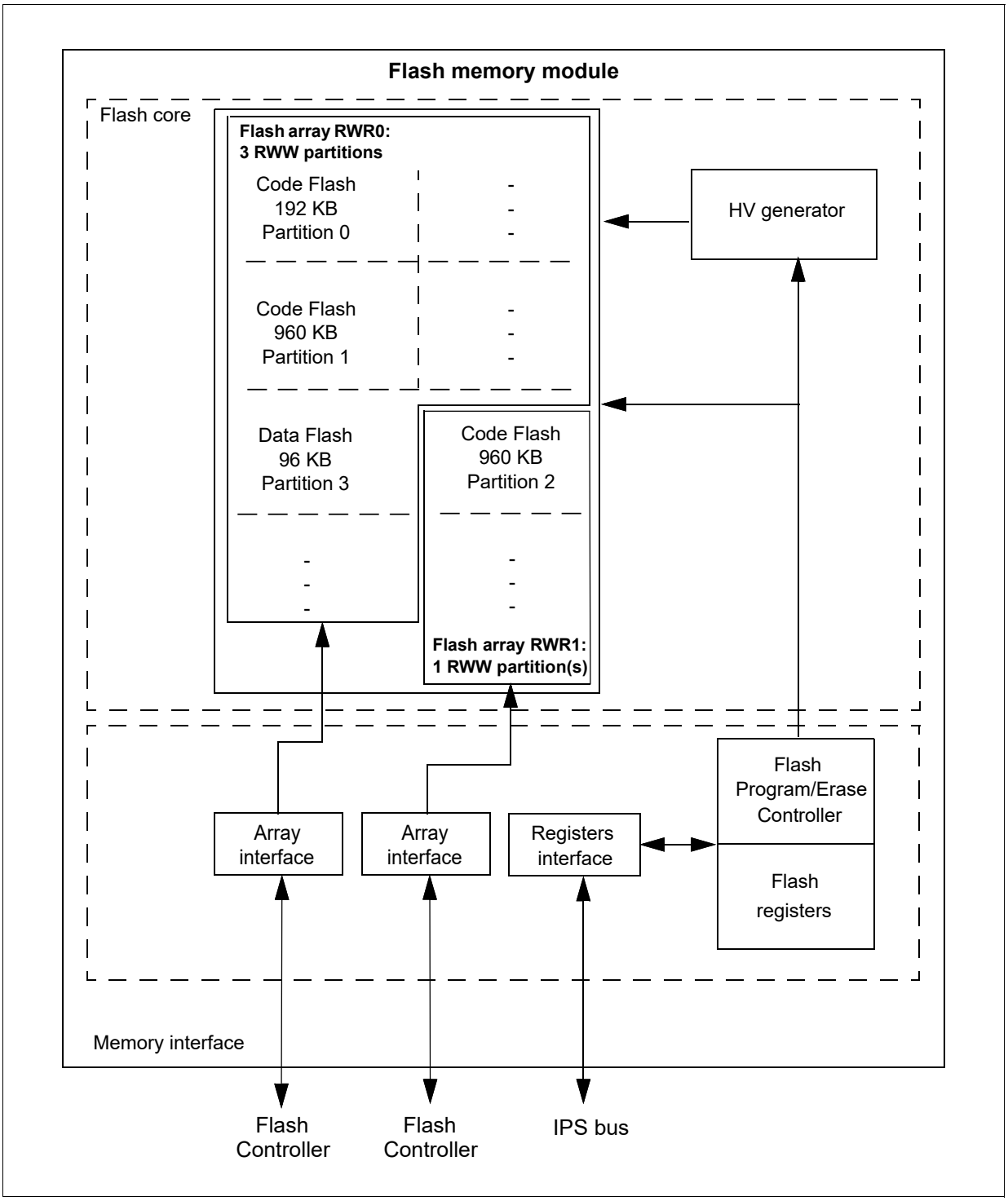
HSM EEPROM emulation is managed as 2 × 16 KB blocks in one partition, referred to as HSM Data flash.

The Modify operations are managed by an embedded flash program/erase controller (FPEC). Commands to the FPEC are delivered through a User Registers interface.

The Read Data bus is 256-wide, while the flash registers are on a separate, 32-bit bus.

The high voltages needed for Program and Erase operations are internally generated and are common for the four partitions.

Figure 130. Flash memory module structure



The flash memory module is designed for use in embedded applications which require high density non-volatile memories with high-speed Read access.

The flash array is addressable:

- For programs in data partition: by word (32 bits) or doubleword (64 bits). ECC granularity is 64 bits.
- For programs in code partition: by word (32 bits) or doubleword (64 bits). ECC granularity is 128 bits.
- For reading code partitions – by page (256 bits).
- For reading data partitions – by half-page (128 bits).

Multiple-word or doubleword writes to the flash memory may be performed to fill the program page buffer (256 bits), enabling page programming (256 bits – requiring 4 doubleword writes) and quad-page programming (1024 bits – requiring 16 doubleword writes). Code flash memory reads always return 256, although Read buffering may be performed by the Bus Interface Unit (BIU). The address for each word retrieved within a page differs from the other addresses in the page by address bits (4:2).

Data flash memory reads always return 128 bits, or two consecutive doublewords of information. The address for each word retrieved within a doubleword differs from the other addresses by address bits (3:2).

The flash memory module supports fault tolerance through Error Correction Code (ECC) and error detection. The ECC implemented within the Data flash memory module corrects up to double bit failures and detects triple bit failures (as uncorrectable).The ECC implemented within the Code flash memory module corrects single bit failures and detects double bit failures (as uncorrectable).

The flash memory module uses an embedded hardware algorithm implemented in the flash program/erase controller to program and erase the flash partitions.

Control logic that works with the software block-enables and software lock mechanisms are included in the embedded hardware algorithm to guard against accidental program/erase. The hardware algorithm performs the necessary steps to ensure that the storage elements are programmed and erased with sufficient margin to guarantee data integrity and reliability.

A programmed bit in the flash array reads as logic level ‘0’ (or low). An erased bit in the flash array reads as logic level ‘1’ (or high).

Programming and erasing of the flash array requires multiple system clock cycles to complete.

The program and erase sequences may be suspended or aborted.

17.1.2 Features

- 30 ns code/data array access time
- Code Partition High Read parallelism: 256 bits
- Data Partition Read parallelism: 128 bits
- Error Correction Code (Double Error Correction-Triple Error Detection) to enhance data integrity
- Data Partition Doubleword program granularity: 64 bits
- Erase with fast erase time at factory
- Blocks one time programmable (OTP) marking
- Read-While-Modify capability when the accesses are to different partitions
- Erase/Program Suspend
- Programming during Erase-Suspend
- Software programmable program/erase protection to avoid unwanted writings
- Temperature range (-40 °C to 150 junction)
- UTEST mode (user-accessible Test modes)
- Alternate interface provided for program and erase operations to dedicated master as private access to blocks
- Independent and Parallel Read access to specific code flash partitions, namely Read-While-Read supported.
- Critical flops, which may affect flash content, are triple-voted for safety application.

17.1.3 Modes of operation

The flash memory module supports the following modes of operation:

- Reset: the flash memory module initializes when the reset is deasserted.
- Read mode: the flash memory module accepts Read accesses to the array.
- Modify mode: it is possible to read and write registers, Interlock Write the memory array, program the memory array, and erase the memory array. Program and Erase operations are initiated by performing array and register writes, and controlled by an internal state machine. Program/Erase operations or User Test modes (Margin Read or Array Integrity Check) are considered Main Modify modes.
- Read-While-Write: The flash memory module accepts Read accesses to array partitions that are not in Modify mode.
- Read-While-Read: The flash memory module accepts parallel Read accesses in separate Read-While-Read partitions.
- Alternate Modify mode: blocks in LOW and MID space could be assigned to Alternate interface, though which it is possible to read and write registers, Interlock Write the assigned memory array, program the assigned memory array, and erase the assigned memory array. Program and Erase operations are initiated by performing array and alternate register writes, and are controlled by the same internal state machine as in Main. Alternate Program/Erase operations are considered Alternate Modify modes. Alternate Modify modes operate concurrently to Main ones and, although functionally independent, timing can be mutually influenced.

Such modes of operation are described in detail in *Section 17.4: Functional description*.

17.2 Flash memory map and description

The flash memory module block subdivision also facilitates independent Erase and Program protection. A software mechanism is provided to independently lock and unlock each block in Low, Mid, High and Data address spaces against program and erase.

17.2.1 Flash array memory map

17.2.1.1 Code flash

Partition 0 accounts for 192 KB of the total flash memory and is divided in 5 user blocks for code storage.

Partition 0 also contains a reserved block named TestFlash in which some One Time Programmable (OTP) user data is stored.

The sectorization of partition 0 of the flash memory is shown in *Table 187*.

Table 187. Flash memory partition 0 memory map

Block	Addresses	Size	Address space	LOCK	SEL
B0F0	0x1800_0000 to 0x1800_7FFF	32 KB	Low	LOWLOCK[7]	LOWSEL[7]
B0F1	0x1800_8000 to 0x1801_7FFF	64 KB	Low	LOWLOCK[8]	LOWSEL[8]
B0F2	0x1801_8000 to 0x1802_7FFF	64 KB	Low	LOWLOCK[9]	LOWSEL[9]
B0F3-UT	0x1FF8_0000 to 0x1FF8_3FFF	16 KB	Low	TSLOCK	—
B0F3-BF	0x1FF0_0000 to 0x1FF0_3FFF	16 KB	Low	LOWLOCK[0]	—
B0F3-TF	—	16 KB	Reserved	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—

The sectorization of partition 1 of the flash memory is shown in *Table 188*:

Table 188. flash memory partition 1 memory map

Block	Addresses	Size	Address space	LOCK	SEL
B1F0	0x0800_0000 to 0x0800_FFFF	64 KB	Low	LOWLOCK[1]	LOWSEL[1]
B1F1	0x0801_0000 to 0x0801_FFFF	64 KB	Low	LOWLOCK[2]	LOWSEL[2]
B1F2	0x0802_0000 to 0x0802_FFFF	64 KB	Low	LOWLOCK[3]	LOWSEL[3]
B1F3	0x0803_0000 to 0x0806_FFFF	256 KB	256K	A256KLOCK[0]	A256KSEL[0]
B1F4	0x0807_0000 to 0x080A_FFFF	256 KB	256K	A256KLOCK[1]	A256KSEL[1]
B1F5	0x080B_0000 to 0x080E_FFFF	256 KB	256K	A256KLOCK[2]	A256KSEL[2]
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—

The sectorization of partition 2 of the flash memory is listed below.

Table 189. Flash memory partition 2 memory map

Block	Addresses	Size	Address space	LOCK	SEL
B2F0	0x080F_0000 to 0x080F_FFFF	64 KB	Low	LOWLOCK[4]	LOWSEL[4]
B2F1	0x0810_0000 to 0x0810_FFFF	64 KB	Low	LOWLOCK[5]	LOWSEL[5]
B2F2	0x0811_0000 to 0x0811_FFFF	64 KB	Low	LOWLOCK[6]	LOWSEL[6]
B2F3	0x0812_0000 to 0x0815_FFFF	256 KB	256K	A256KLOCK[3]	A256KSEL[3]
B2F4	0x0816_0000 to 0x0819_FFFF	256 KB	256K	A256KLOCK[4]	A256KSEL[4]
B2F5	0x081A_0000 to 0x081D_FFFF	256 KB	256K	A256KLOCK[5]	A256KSEL[5]
—	—	—	—	—	—
—	—	—	—	—	—

17.2.1.2 Data flash

Data flash is allocated in partition 3 of the flash memory as 4 × 16 KB blocks destined for data storage.

The data flash sectorization of the flash memory module is listed below.

Table 190. Flash memory EEPROM data partition 3 memory map

Block	Addresses	Size	Address space	LOCK	SEL
D0_00	0x08F0_0000 to 0x08F0_3FFF	16 KB	High	HIGHLOCK[0]	HIGHSEL[0]
D0_01	0x08F0_4000 to 0x08F0_7FFF	16 KB	High	HIGHLOCK[1]	HIGHSEL[1]
D0_02	0x08F0_8000 to 0x08F0_BFFF	16 KB	High	HIGHLOCK[2]	HIGHSEL[2]
D0_03	0x08F0_C000 to 0x08F0_FFFF	16 KB	High	HIGHLOCK[3]	HIGHSEL[3]

HSM Data Flash is allocated in partition 3 and is composed by 2 x 16 KB blocks. The HSM Data Flash sectorization is listed below.

Table 191. Flash memory HSM EEPROM data partition 3 memory map

Block	Addresses	Size	Address space	LOCK	SEL
D0_04	0x18F0_0000 to 0x18F0_3FFF	16 KB	Mid	MIDLOCK[0]	MIDSEL[0]
D0_05	0x18F0_4000 to 0x18F0_7FFF	16 KB	Mid	MIDLOCK[1]	MIDSEL[1]

17.2.1.3 TestFlash and UTEST block memory map

The TestFlash physical block contains OTP user sections: UTEST (16 KB) and BAF (16 KB) plus a Manufacturer Reserved Test Flash Area (16 KB) which is referred to as TestFlash logical block.
The TestFlash logical block exists outside the normal address space. The UTEST block is programmed and read independently of the other blocks. UTEST is included to support systems which require nonvolatile memory for security and to store system initialization information.

The usage of reserved UTEST block is detailed in *Table 192*.

Table 192. UTEST block memory map

Address Offset	Description	Size	Access
UTEST			
0x0000	Reserved	32 bytes	R
0x0020	Factory Erase Diary Location	16 bytes	R/P
0x0030	Test Mode Disable Override Passcode	16 bytes	R/P
0x0040	Test Mode Disable Seal	16 bytes	R/P
0x0050	Test Mode Disable Group A	16 bytes	R/P
0x0060	Test Mode Disable Group B	16 bytes	R/P
0x00A0	Unique chip identifier (UID)	32 bytes	R
0x00C0	Reserved	16 bytes	R
0x00D0	Reserved	48 bytes	R
0x0100	Password to enable the Flash Test Mode when in FA	32 bytes	R

Table 192. UTEST block memory map (continued)

Address Offset	Description	Size	Access
0x0120	JTAG Password	32 bytes	R/P
0x0140(1)	PASS password - Group 0	32 bytes	R/P
0x0160	PASS password - Group 1	32 bytes	R/P
0x0180	PASS password - Group 2	32 bytes	R/P
0x01A0	PASS password - Group 3	32 bytes	R/P
0x01C0	Reserved	32 bytes	R/P
0x01E0	Life Cycle slot 0 - ST Production	32 bytes	R/P
0x0200	Life Cycle slot 1 - Customer Delivery	32 bytes	R/P
0x0220	Life Cycle slot 2 - OEM Production	32 bytes	R/P
0x0240	Life Cycle slot 3 - In Field	32 bytes	R/P
0x0260	Life Cycle slot 4 - Failure Analysis	32 bytes	R/P
0x0280	Customer Single Bit Correction area	32 bytes	R
0x02A0	Reserved	32 bytes	R
0x02C0	Customer Uncorrectable Bit Detection area	32 bytes	R
0x02E0	Customer Programmable Detection Area	32 bytes	R
0x0300	DCF Start record	8 bytes	R/P
0x0308	Reserved	144 bytes	R/P
0x0398	UTEST DCF Records	3176 bytes	R/P
0x1000	BAF Soft DCF Records	1024 bytes	R/P
0x1400	Customer OTP data	9216 bytes	R/P
0x3800	OTA Signature area	2048 bytes	R/P

1. The offset 0x140 contains the least significant 32-bit of the PASS password - Group 0. The offset 0x15C contains the most significant 32-bit of the PASS password - Group 0. The same endianess also applies to the other groups of the PASS password.

UTEST block supports RWW, and is grouped with the Code Flash in partition 0.

Programming of any section of the UTEST block is subject to similar restrictions as the array in terms of how ECC is calculated. Only one program is allowed per 64-bit ECC segment.

The UTEST nonvolatile memory section is OTP and erase is not allowed. User mode programming of the UTEST section is enabled only when MCR.PEAS is high. The UTEST section is divided into a *Flexible* area where content is managed at System level, and a *Reserved* area where addresses are pre-allocated for storing certain content. The UTEST section may be locked and unlocked against programming via LOCK0.TSLOCK in conjunction with the relative sideband (System specific). For locking purposes, UTEST is treated as a stand-alone block even though it resides in the TestFlash block. UTEST access (both Read and Modify) is disabled in Test mode once the Test Mode Disable Seal password is written. Test Flash logical block gets also protected by the Test Mode Disable Seal.

The UTEST section is also set as OPP once the Test Mode Disable Seal password is written, thus programming is only allowed on virgin 64-bit locations.

The other area of the TestFlash physical block is BAF and may be used for defined functions (possibly to store boot code, other configuration words or factory process codes). This nonvolatile memory area is OTP and cannot be erased once programmed.

Although the BAF (like UTEST) is part of the TestFlash physical block, OPP functionality is managed through sideband (SoC specific). Program protection is also managed through sideband (SoC specific), in conjunction with LOCK0.LOWLOCK. For locking and OPP purposes, BAF is treated as a stand-alone block.

When the UTEST section (via the entire TestFlash block) is protected via Test Mode Disable Seal, the BAF is also protected from any high voltage operation which may alter the content. Access to BAF is read only for FPEC in the event that the test interface is open (Failure Analysis scenario).

Customer Detection and Correction area in UTEST are related to error injection for safety reaction for Code Memory only.

17.2.1.4 RWR1_UTEST virtual block memory map

RWR1_UTEST virtual block is an area allocated for the purpose of injecting safety relevant error and to test the reaction path. This task is accomplished in the primary Read-While-Read partition 0 directly within UTEST memory map. RWR1_UTEST virtual block is assigned for the second, parallel, Read-While-Read partition 1 (see *Section 17.4.2.1: Read While Read*).

Warning: Customer Programmable Detection Area for RWR1_UTEST virtual block is qualified by UT0.CPA and UT0.CPE. UT0.CPR has no effect.

Table 193. RWR1_UTEST virtual block memory map

Address Offset	Description	Size	Access
RWR1_UTEST (Base Address 0x1FFBC000)			
0x0000	Reserved	640 bytes	
0x0280	Customer Single Bit Correction area	32 bytes	R
0x02A0	Reserved	32 bytes	R
0x02C0	Customer Uncorrectable Bit Detection area	32 bytes	R
0x02E0	Customer Programmable Detection Area	32 bytes	R

17.3 Register memory maps and descriptions

In this section some nonvolatile registers are described as well as the usual volatile registers. Note that such entities are not Flip-Flops, but Test Flash block locations with a specific purpose.

During the flash memory initialization phase, the FPEC reads these nonvolatile registers and updates their corresponding volatile registers. When the FPEC detects ECC uncorrectable errors in these special locations, it behaves in the following way:

- In the case of failing system locations (configurations, redundancy, EmbAlgo firmware), the initialization phase is interrupted and a Fatal Error is flagged.
- In the case of failing coherency checks, the volatile registers are filled with ‘1’s and the Flash Initialization ends setting MCR.RE.

17.3.1 Register memory maps

This section presents the memory mapping of all the flash memory registers.

17.3.1.1 User register memory maps

The Flash User registers represent the communication interface between the host CPU and the FPEC.

Some register bits (command bits) are Read/Write for the CPU and read only for the FPEC. Other register bits (status bits) are Read/Write for the FPEC and read only for the CPU.

The Flash User registers mapping is shown in *Table 194*.

Table 194. Flash user registers memory map

Address offset	Description	Location
0x0000	Module Configuration register (MCR)	*Section 17.3.2.1*
0x0004	Alternate Module Configuration register (MCRA)	*Section 17.3.2.2*
0x0008	Extended Module Configuration register (MCRE)	*Section 17.3.2.3*
0x000C–0x000F	Reserved	
0x0010	Locking 0 register (LOCK0)	*Section 17.3.2.4*
0x0014	Locking 1 register (LOCK1)	*Section 17.3.2.5*
0x0018	Locking 2 register (LOCK2)	*Section 17.3.2.6*
0x001C	Locking 3 register (LOCK3)	*Section 17.3.2.7*
0x0020–0x0027	Reserved	
0x0028	Alternate Locking 0 register (LOCK0A)	*Section 17.3.2.8*
0x002C	Alternate Locking 1 register (LOCK1A)	*Section 17.3.2.9*
0x0038	Select 0 register (SEL0)	*Section 17.3.2.10*
0x003C	Select 1 register (SEL1)	*Section 17.3.2.11*
0x0040	Select 2 register (SEL2)	*Section 17.3.2.12*
0x0044	Select 3 register (SEL3)	*Section 17.3.2.13*
0x0048	Module Configuration register 2 (MCR2)	*Section 17.3.2.14*
0x004C	Program Address register (PAR)	*Section 17.3.2.15*
0x0050	Address register (ADR)	*Section 17.3.2.16*
0x0054	User Test register 0 (UT0)	*Section 17.3.2.17*
0x0058	User Multiple Input Signature register 0 (UM0)	*Section 17.3.2.18.1*

Table 194. Flash user registers memory map (continued)

Address offset	Description	Location
0x005C	User Multiple Input Signature register 1 (UM1)	*Section 17.3.2.18.2*
0x0060	User Multiple Input Signature register 2 (UM2)	*Section 17.3.2.18.3*
0x0064	User Multiple Input Signature register 3 (UM3)	*Section 17.3.2.18.4*
0x0068	User Multiple Input Signature register 4 (UM4)	*Section 17.3.2.18.5*
0x006C	User Multiple Input Signature register 5 (UM5)	*Section 17.3.2.18.6*
0x0070	User Multiple Input Signature register 6 (UM6)	*Section 17.3.2.18.7*
0x0074	User Multiple Input Signature register 7 (UM7)	*Section 17.3.2.18.8*
0x0078	User Multiple Input Signature register 8 (UM8)	*Section 17.3.2.18.9*
0x007C	User Multiple Input Signature register 9 (UM9)	*Section 17.3.2.18.10*
0x0080	Over Program Protection 0 (OPP0)	*Section 17.3.2.19.1*
0x0084	Over Program Protection 1 (OPP1)	*Section 17.3.2.19.2*
0x0088	Over Program Protection 2 (OPP2)	*Section 17.3.2.19.3*
0x008C	Over Program Protection 3 (OPP3)	*Section 17.3.2.19.4*
0x0090	Test Mode Disable Password Check (TMD)	*Section 17.3.2.20*
0x0094	HSM ALT-SEL 0 register (ALTSEL0)	*Section 17.3.2.21.1*
0x0098	HSM ALT-SEL 1 register (ALTSEL1)	*Section 17.3.2.21.2*
0x0100	Erase Lock Protection 0 (ELOCK0)	*Section 17.3.2.22.1*
0x0104	Erase Lock Protection 1 (ELOCK1)	*Section 17.3.2.22.2*
0x0108	Erase Lock Protection 2 (ELOCK2)	*Section 17.3.2.22.3*
0x010C	Erase Lock Protection 3 (ELOCK3)	*Section 17.3.2.22.4*
0x0110	Program Lock Protection 0 (PLOCK0)	*Section 17.3.2.23.1*
0x0114	Program Lock Protection 1 (PLOCK1)	*Section 17.3.2.23.2*
0x0118	Program Lock Protection 2 (PLOCK2)	*Section 17.3.2.23.3*
0x011C	Program Lock Protection 3 (PLOCK3)	*Section 17.3.2.23.4*
0x0134	Parallel Address register (ADR2)	*Section 17.3.2.24*

17.3.2 User register descriptions

This section consists of user register descriptions in address order.

17.3.2.1 Module Configuration register (MCR)

The Module Configuration register is used to enable and monitor all the Modify operations of the flash memory module and to monitor the Read operations of Read-While-Read Interface 0.

Address: 0x0000 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	RRE	AEE	EEE	DWEE				0	0	0	SBC1	SAK	LSW	PEP	PES
W		w1c	w1c	w1c								w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EER	RWE	SBC	RE	PEAS	DONE	PEG	PECIE	FERS	0	0	PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 131. Module Configuration register (MCR)

Table 195. MCR field descriptions

Field	Description
30 RRE	Read Reference Error RRE provides information on previous reads monitoring the analog Read references. If the current and voltage Read references are out of range, this bit is set to signify that previous reads requested may have been corrupted. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A Read Reference Error occurred during a previous Read.
29 AEE	Address Encode Error On every Read request to the flash memory, the incoming address is compared to an encoded address (row, column, and block) from the memory array using the read data sense amp timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error is recorded. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 An Address Encode Error occurred during a previous Read.
28 EEE	EDC after ECC Error EEE provides information on previous reads monitoring the EDC after ECC feature. On every Read request to the flash memory, ECC parity is recomputed (encoding) just after ECC decoding, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error is reported. The EEE flag indicates this event when data is legal or containing up to 3 bit errors. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A previous Read may have been decoded by an ECC logic corrupted.

Table 195. MCR field descriptions (continued)

Field	Description
27:24 DWEE	Double Word Uncorrectable ECC Error DWEE identifies which Double Word is involved in ECC Uncorrectable Error Event. In Code Flash 4 Double Words are read in parallel and ECC is evaluated on a couple of Double Words basis. Therefore DWEE value could be: 1111 if an ECC Uncorrectable Error is found in both couples of Double Words 1100 if an ECC Uncorrectable Error is found in the couple of Double Words at high address 0011 if an ECC Uncorrectable Error is found in the couple of Double Words at low address 0000 if no ECC Uncorrectable Error is found In Data Flash 2 Double Words are read in parallel and ECC is evaluated on a single Double Word basis. Therefore DWEE value could be: 1111 if an ECC Uncorrectable Error is found in both Double Words 1010 if an ECC Uncorrectable Error is found in Double Word at high address 0101 if an ECC Uncorrectable Error is found in Double Word at low address 0000 if no ECC Uncorrectable Error is found DWEE is a read only. Flags bits get cleared when EER is cleared.
20 SBC1	Single Bit Correction 1 SBC1 provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection or ECC Double Error correction, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE1). 0 Reads are occurring normally. 1 An ECC Single Error occurred and was corrected during a previous Read. In Code Flash, with 128b ECC coding, ECC correction is evaluated on couple of Double word basis. In Data Flash for EEE, ECC correction is evaluated on each Double word.
19 SAK	Suspend Acknowledge SAK provides information about the Acknowledge of a suspend request at the setting of DONE: SAK qualifies if the suspend has been accepted (initiated by PSUS or ESUS setting) or if the running command has been completed, since very close to natural end. The aim of this bit is for monitor. The sequence to resume normal functionality (PSUS or ESUS clearing) is the same both is suspend has been acknowledged or not. SAK is read Only 0 Operation has been completed prior that Suspend got processed 1 Suspend Acknowledged
18 LSW	Lock-Sel Write Error LSW provides information about Lock and Sel write denied during the program and erase sequence. If LSW is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Lock and Sel write successful 1 Lock and Sel write not succeeded

Table 195. MCR field descriptions (continued)

Field	Description
17 PEP	Program and Erase Protection Error PEP provides information about program and erase operations with respect to protection errors. A protection error occurs if a program is attempted to a Locked block, or if an erase is selected to a locked block. This is evaluated prior to the operation beginning, and if an error is detected, high voltage operations will not be attempted for this request. If PEP is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program and Erase protection errors do not exist 1 Previous program or erase protection error encountered
16 PES	Program Erase Sequence Error PES provides information about program and erase operations with respect to sequence errors. A sequence error occurs when the program or erase sequence is not followed exactly. If an out-of-order event is detected, PES will assert, and the remainder of the sequence will not be accepted. Sequence errors are checked from the time the PGM or ERS are set (or remains set from previous operation) until EHV is set. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program or Erase phase accepted 1 Program or Erase command encountered an error
15 EER	ECC uncorrectable Event Error EER provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. EER may not be set to ‘1’ by the user. EER is not set for a ECC correctable Error detection and related correction event. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 An ECC uncorrectable Error occurred during a previous Read. In Code Flash, with 128b ECC coding, ECC error is evaluated on couple of Double word basis. In Data Flash for EEE, ECC error is evaluated on each Double word.
14 RWE	Read-while-Write event Error RWE provides information on previous Reads during a Modify operation. Read-While-Write Error means that a Read access to the Flash Array partition has occurred while the FPEC was performing a Program or Erase operation or an Array Integrity Check on the same partition. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A RWW Error occurred during a previous Read.

Table 195. MCR field descriptions (continued)

Field	Description
13 SBC	double Bit Correction SBC provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE). 0 Reads are occurring normally. 1 An ECC Double Error occurred and was corrected during a previous Read. In Data Flash for EEE, ECC correction is evaluated on each Double word.
12 RE	Reset Error This bit provides information on previous resets. Checks are done within the flash memory, and if a coherency issue or ECC error is detected during the reset reads used for trimming on the flash memory, this status flag will assert. This bit is status only, and is not writable. 0 Flash Boot phase has occurred normally. 1 A reset error has been encountered.
11 PEAS	Program/Erase Access Space PEAS is used to indicate which space is valid for Program and Erase operations: main array space or UTEST section within Test Flash block. – PEAS is captured and held with the first Interlock Write done for Modify operations. – PEAS is retained between sampling events (that is subsequent to first interlock writes). – PEAS is changed during erase-suspended program (when program is in UTEST) and reverts back to its original state (related to erase suspended operation) once the erase-suspended program is completed (at MCR.PGM 1-0 transition). 0 UTEST address space is disabled for program and main address space enabled. 1 UTEST address space is enabled for program and main address space disabled.
10 DONE	Modify operation DONE DONE indicates if the flash memory module is performing a high voltage operation. DONE is set to ‘1’: – On termination of the flash memory module reset. – At the end of program and Erase High Voltage sequences. – After “1-0” transition of EHV, which aborts a high voltage Erase/Program operation (within a time equal or below to t_{ESUS}/t_{PSUS} that are Erase/Program Suspend Latency). – After “0-1” transition of ESUS/PSUS which suspends an Erase/Program operation (within t_{ESUS}/t_{PSUS} time equal to the Erase/Program Suspend Latency). DONE is cleared to ‘0’: – just after “0-1” transition of EHV, which initiates a high voltage operation, or after resuming a suspended operation. 0 Flash memory is executing a high-voltage operation. 1 Flash memory is not executing a high-voltage operation.

Table 195. MCR field descriptions (continued)

Field	Description
9 PEG	Program/Erase Good PEG is automatically updated to show the completion status of the last Flash Program or Erase sequence involving high voltage operations. PEG is evaluated on the completion status of only the interlocked doubleword during page program or quad-page programming where some doubleword addresses are not interlocked (already programmed or intended to remain virgin). PEG is set to ‘0’ after Aborting a Program/Erase high voltage operation, indicating sequence failure. PEG is set to ‘1’ when the flash memory module is reset, unless a Flash Initialization error has been detected. **Note:** PEG is valid only when PGM=1 or ERS=1, or both, and after DONE 0-1 transitions due to an Abort or the completion of a Program/Erase operation. PEG remains valid until PGM/ERS 1-0 transitions or EHV 0-1 transitions. PEG is not valid after DONE 0-1 transitions when ESUS/PSUS is set to logic ‘1’. If Program or Erase is attempted on blocks that are locked, PEG=1 indicating that the content of the block were properly protected from the Program or Erase operation. Blocks can be locked through LOCK0/1/2/3 registers OR with flh_plock0/1/2/3 sideband signals. If a Program operation tries to write ‘1’ to bits that are ‘0’, the program operation is correctly executed on the new bits to be programmed at ‘0’, but PEG is cleared indicating that the requested operation has failed. If Programming during an Erase-Suspend a location that was under suspended erase, program is not executed and PEG is set to ‘0’. With end2end ECC scheme, during an Interlock Write also ECC parity bits are input. After setting EHV, the FPEC checks the integrity of the data received in terms of ECC. If any doubleword in the doubleword program, in the page program or in the quad-page program, contains an inconsistency between data and parity, the whole operation is rejected and a PEG=0 is set. PEG behavior with blocks marked as OPP (though flh_otpen0/1/2/3 sideband signals) is the following: PEG=0 if the location to be programmed is not virgin, and thus would not be over programmed. In Array Integrity Check or Margin Read PEG is not altered when the operation is completed, regardless of the occurrence of any error, abort, break point or suspend. The presence of errors can only be detected by comparing the checksum value stored in UM0-9 and by checking the MCR flags. 0 Program, Erase operation failed or Program, Erase aborted. 1 Program or Erase operation successful.
8 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag. 0 An interrupt request is not generated when the DONE flag is set. 1 An interrupt request is generated when the DONE flag is set.

Table 195. MCR field descriptions (continued)

Field	Description
7 FERS	Factory Mode FERS is used for Erase and Program operations to enable a faster operation. Factory Mode must adhere to *Initial maximum (All Temperatures)* conditions, with respect to voltage conditions and Write/Erase cycling. The temperature can range from -40 °C to 150 °C. Factory Mode does not support Read and RWW operations. – FERS may be set to enable a Factory Erase or Program after setting ERS or PGM but prior to setting EHV. – FERS can be cleared only when EHV is low and DONE is high. – FERS is cleared on reset **Note:** Factory Mode may be permanently disabled by writing a location in the UTEST Nonvolatile Memory space. Offset 0x0020 in the UTEST Nonvolatile Memory space is checked at reset, and if programmed with data that contains at least one zero, a Factory Mode is not allowed, and FERS is locked. Writes attempted that violate the above cases result in FERS remaining cleared. 0 Factory Mode is disabled. 1 Factory Mode is enabled.
4 PGM	Program PGM is used to setup the flash memory module for a Program operation. – “0-1” transition of PGM initiates a program sequence. – “1-0” transition of PGM ends the program sequence. PGM can only be set in the following conditions: – User Mode Read (ERS is low and UT0.UTE is low); – Erase Suspend (ERS and ESUS high) with EHV low. PGM can only be cleared by the user when PSUS and EHV are low and DONE is high. PGM is cleared on reset. **Note:** In an erase-suspended program, programming Flash memory locations in blocks which were being operated on in the Erase is forbidden. Command is not accepted and PEG is set to ‘0’ 0 The flash memory is not executing a program sequence. 1 The flash memory is executing a program sequence.

Table 195. MCR field descriptions (continued)

Field	Description
3 PSUS	Program Suspend PSUS is used to indicate that the flash memory module is in Program Suspend or in the process of entering a Suspend state. The flash memory module is in Program Suspend when PSUS=1 and DONE=1. – PSUS can be set high only when PGM and EHV are high. – “0-1” transition of PSUS starts the sequence which sets DONE and places the Flash in Program Suspend. The flash memory module enters Suspend within t_{PSUS} of this transition. – PSUS can be cleared only when DONE and EHV are high. – “1-0” transition of PSUS with EHV=1 starts the sequence which clears DONE and returns the module to Program. The flash memory module cannot exit Program Suspend and clear DONE while EHV is low. In the case of Program Suspend while in Erase Suspend (PSUS=1, ESUS=1), PSUS has to be cleared first in order to resume currently higher level suspended operation that is program. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.
2 ERS	Erase ERS is used to setup the flash memory module for an Erase operation. – “0-1” transition of ERS initiates an Erase sequence. – “1-0” transition of ERS ends the Erase sequence. ERS can be set only under User Mode Read (PGM is low and UT0.UTE is low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 The flash memory is not executing an Erase sequence. 1 The flash memory is executing an Erase sequence.

Table 195. MCR field descriptions (continued)

Field	Description
1 ESUS	Erase Suspend ESUS is used to indicate that the flash memory module is in Erase Suspend or in the process of entering a Suspend state. The flash memory module is in Erase Suspend when ESUS = 1 and DONE=1. – ESUS can be set high only when ERS and EHV are high and PGM is low. – “0-1” transition of ESUS starts the sequence which sets DONE and places the Flash in Erase Suspend. The flash memory module enters Suspend within t_{ESUS} of this transition. – ESUS can be cleared only when DONE and EHV are high and PGM is low. – “1-0” transition of ESUS with EHV=1 starts the sequence which clears DONE and returns the module to Erase. The flash memory module cannot exit Erase Suspend and clear DONE while EHV is low. In case of Program Suspend while in Erase Suspend (PSUS=1, ESUS=1), ESUS cannot be cleared first. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.

Table 195. MCR field descriptions (continued)

Field	Description
0 EHV	Enable High Voltage The EHV bit enables the flash memory module for a high voltage Program/Erase operation. EHV is cleared on reset. EHV must be set after an Interlock Write to start a Program/Erase sequence. EHV may be set under one of the following conditions: – Erase (ERS=1, ESUS=0, UT0.AIE=0) – Program (ERS=0, ESUS=0, PGM=1, UT0.AIE=0) – Erase-suspended program (ERS=1, ESUS=1, PGM=1, PSUS=0, UT0.AIE=0). For a program operation to initiate while an Erase is suspended, EHV must be cleared while in Erase-Suspend (after DONE transitions to 1) before setting PGM. In normal operation, a “1-0” transition of EHV with DONE high, PSUS low and ESUS low terminates the current Program/Erase high voltage operation. In an erase-suspended program operation, a “1-0” transition of EHV with DONE, ESUS, PGM and ERS high and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a “1-0” transition of EHV with DONE low and the eventual Suspend bit for the current program/erase operation low. An Abort causes the value of PEG to be cleared, indicating failed Program/Erase. Address locations being operated on by the aborted operation contain indeterminate data after an Abort. EHV cannot be set to ‘1’ after an Abort request (EHV being cleared) until the Abort sequence is completed (DONE transitions to 1). Once the Abort sequence is completed (DONE=1) a new operation of the kind aborted can start provided that a new interlock is given and then EHV is set to ‘1’ again. A suspended operation cannot be aborted. Aborting a high voltage operation leaves the flash memory module addresses in an indeterminate data state. This may be recovered by executing an Erase on the affected blocks. EHV may be written during Suspend. EHV must be high to exit Suspend, resuming previously suspended command. EHV cannot not be written during transition into Suspend state (after Suspend bit (ESUS or PSUS) is set high) until DONE transitions to ‘1’. EHV may not be written after ESUS is set and before DONE transitions high. EHV may not be cleared after ESUS is cleared and before DONE transitions low. 0 The flash memory is not enabled to perform a high voltage operation. 1 The flash memory is enabled to perform a high voltage operation.

A number of MCR bits are protected against Write when another bit, or set of bits, are in a specific state. These Write locks are covered on a bit-by-bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

The flash memory module does not allow the user to write bits simultaneously (which puts the device into an illegal state). This is implemented through a priority mechanism among the bits shown in *Table 196*.

Table 196. MCR Bits Set/Clear Priority Levels

Priority level	MCR bits
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If the user attempts to write two or more MCR bits simultaneously then only the bit with the lowest priority level is written.

17.3.2.2 Alternate Module Configuration register (MCRA)

The Alternate Module Configuration register is used to enable and monitor all the Modify operations of the flash memory module managed through Alternate interface.

Address: 0x0004 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	SAKA	LSWA	PEPA	PESA
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	PEASA	DONEA	PEGA	0	0	0	0	PGMA	PSUSA	ERSA	ESUSA	EHVA
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 132. Alternate Module Configuration register (MCRA)

Table 197. MCRA field descriptions

Field	Description
19 SAKA	Alternate Suspend Acknowledge SAKA provides information about the Acknowledge of a suspend request at the setting of DONEa: SAKA qualifies if the suspend has been accepted (initiated by PSUSa or ESUSa setting) or if the running command has been completed, since very close to natural end. The aim of this bit is for monitor. The sequence to resume normal functionality (PSUSa or ESUSa clearing) is the same both is suspend has been acknowledged or not. SAKA is read Only 0 Operation has been completed prior that Suspend got processed. 1 Suspend Acknowledged
18 LSWA	Alternate Lock Write Error LSWA provides information about Lock write denied during the program and erase sequence. If LSWA is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Lock and Sel write successful. 1 Lock and Sel write not succeeded.
17 PEPA	Alternate Program and Erase Protection Error PEPA provides information about program and erase operations with respect to protection errors. A protection error occurs if a program is attempted to a Locked block, or if an erase is selected to a locked block. This is evaluated prior to the operation beginning, and if an error is detected, high voltage operations will not be attempted for this request. If PEPA is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program and Erase protection errors do not exist. 1 Previous program or erase protection error encountered.
16 PESA	Alternate Program Erase Sequence Error PESA provides information about program and erase operations with respect to sequence errors. A sequence error occurs when the program or erase sequence is not followed exactly. If an out-of-order event is detected, PESA will assert, and the remainder of the sequence will not be accepted. Sequence errors are checked from the time the PGMA or ERSA are set (or remains set from previous operation) until EHVA is set. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program or Erase phase accepted 1 Program or Erase command encountered an error
11 PEASA	Program/Erase Access Space PEASA is used to indicate which space is valid for Program and Erase operations: main array space or UTEST block. It should be noted that erase is not allowed to the UTest NVM space, therefore If an interlock write is done to the UTest NVM space with ERSA set, PEASA will remain 0. See PEAS definition in the MCR register for more information(*Table 195*).
10 DONEA	Modify operation DONE DONEA indicates if the embedded flash memory is performing a high-voltage operation for the Alternate program and erase interface. See DONE definition in the MCR register for more information (*Table 195*).

Table 197. MCRA field descriptions (continued)

Field	Description
9 PEGA	Program/Erase Good The PEGA bit indicates the completion status of the last Flash memory program or erase sequence for which high-voltage operations were initiated for the Alternate program and erase interface. See PEG definition in the MCR register for more information (*Table 195*). It should be noted that in the event of a error on the alternate interface PEGA will clear, but the ADR register will not be updated with the failing location. The ADR register is only valid for main interface program or erase fails, and other fails noted in the ADR register.
4 PGMA	Program PGMA is used to set up embedded flash memory for a program operation for the Alternate program and erase interface. See PGM definition in the MCR register for more information (*Table 195*).
3 PSUSA	Program Suspend PSUSA is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state for the Alternate program and erase interface. See PSUS definition in the MCR register for more information (*Table 195*).
2 ERSA	Erase ERSA is used to set up embedded flash memory for an erase operation for the Alternate program and erase interface. See ERS definition in the MCR register for more information (*Table 195*).
1 ESUSA	Erase Suspend ESUSA is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state for the Alternate program and erase interface. See ESUS definition in the MCR register for more information (*Table 195*).
0 EHVA	Enable High Voltage The EHVA bit enables the embedded flash memory for a high-voltage program/erase operation for the Alternate program and erase interface. See EHV definition in the MCR register for more information (*Table 195*).

A number of MCR bits are protected against Write when another bit, or set of bits, are in a specific state. These Write locks are covered on a bit-by-bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

17.3.2.3 Extended Module Configuration register (MCRE)

The Extended Module Configuration register value is based on the size, blocks and address space configuration of the flash memory module.

Address: 0x0008 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		N128KL		N256K					N64KH			N32KH		N16KH		
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	N64KM			N32KM		N16KM			N64KL			N32KL		N16KL		
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1

Figure 133. Extended Module Configuration register (MCRE)

Table 198. MCRE field descriptions

Field	Description
30:29 N128KL	Number of 128 KB blocks in low space. 00 Zero 128 KB block. 01 One 128 KB block. 10 Two 128 KB blocks. 11 Three 128 KB blocks.
28:24 N256K	Number of 128 KB and 256 KB blocks in the 256K space The value of the N256K field is dependent upon the size of the embedded flash memory. N256K is read only. N256K value represents in the number of blocks effectively available. 00000 Zero block. 00001 One block. 00010 Two blocks. 00011 Three blocks. 00100 Four blocks. 00101 Five blocks. 00110 Six blocks. 00111 Seven blocks 01000 Eight blocks. 01001 Nine blocks. 01010 Ten blocks. … 11111 thirty-one blocks.
23:21 N64KH	Number of 64 KB blocks in high space N64KH is read only. 000 Zero 64 KB block. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.

Table 198. MCRE field descriptions (continued)

Field	Description
20:19 N32KH	Number of 32 KB blocks in high space N32K is read only. 00 Zero 32 KB block. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
18:16 N16KH	Number of 16 KB blocks in high space N16K is read only. 000 Zero 16 KB block. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
15:13 N64KM	Number of 64 KB blocks in mid space N64KM is read only. 000 Zero 64 KB block. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
12:11 N32KM	Number of 32 KB blocks in mid space N32KM is read only. 00 Zero 32 KB block. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
10:8 N16KM	Number of 16 KB blocks in mid space N16KM is read only. 000 Zero 16 KB block. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.

Table 198. MCRE field descriptions (continued)

Field	Description
7:5 N64KL	Number of 64 KB blocks in low space N64KL is read only. 000 Zero 64 KB block. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Three 64 KB blocks + One 96KB block. 110 Reserved. 111 Reserved.
4:3 N32KL	Number of 32 KB blocks in low space N32KL is read only. 00 Zero 32 KB block. 01 Two 32 KB blocks. 10 One 32 KB block. 11 Three 32 KB blocks.
2:0 N16KL	Number of 16 KB blocks in low space N16KL is read only. 000 Zero 16 KB block. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Three 16 KB blocks. 110 Five 16 KB blocks. 111 Reserved.

17.3.2.4 Lock 0 register (LOCK0)

The LOCK0 (Low/Mid Address Space Block Locking) register provides a means of protecting blocks from being modified. LOWLOCK and MIDLOCK bits corresponding to non existent blocks are read as ‘1’.

To determine the effective status of the Low/Mid Address Space, combine (Or) LOCK0 with sidebands:

- flh_plock0 – in order to evaluate protection against programming.
- flh_elock0 – in order to evaluate protection against erase.

The effect of programming or erasing a block protected with LOCK0 is that no modification occurs and PEG is set to ‘1’.

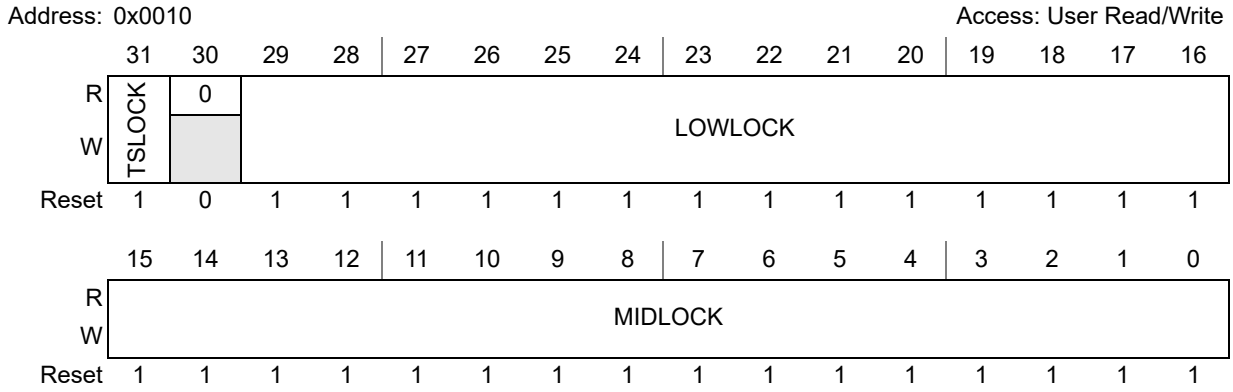


Figure 134. Lock 0 register (LOCK0)

Table 199. LOCK0 field descriptions

Field	Description
31 TSLOCK	UTEST NVM block lock This bit is used to lock the UTEST NVM from programming (UTEST NVM is not erasable). The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the TSLOCK bit is locked. The effective status of the UTEST protection is done combining (ORed) LOCK0.TSLOCK with sidebands *flh_plock0.31* in order to evaluate protection against program. 0 Test Address Space Block is unlocked and can be modified. 1 Test Address Space Block is locked and cannot be modified.

Table 199. LOCK0 field descriptions (continued)

Field	Description
29:16 LOWLOCK	Low address space block lock These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the LOWLOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the LOWLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Low Address Space Block is unlocked and can be modified. 1 Low Address Space Block is locked and cannot be modified.
15:0 MIDLOCK	Mid address space block lock These bits are used to lock the blocks of Mid Address Space from Program and Erase. The block numbering for the Mid Blocks starts at MIDLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the MIDLOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the MIDLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Mid Address Space Block is unlocked and can be modified. 1 Mid Address Space Block is locked and cannot be modified.

17.3.2.5 Lock 1 register (LOCK1)

The LOCK1 (High/Data Address Space Block Locking) register provides a means of protecting blocks from being modified. HIGHLOCK bits corresponding to nonexistent blocks are read as ‘1’.

- To determine the effective status of the High Address Space, combine (Or) LOCK1 with sidebandsflh_plock1 – in order to evaluate protection against program
- flh_elock1 – in order to evaluate protection against erase.

The effect of programming/erasing a block protected with LOCK1 is that no modification occurs and PEG is set to ‘1’.

Address: 0x0014 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HIGHLOCK															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 135. Lock 1 register (LOCK1)

Table 200. LOCK1 field descriptions

Field	Description
15:0 HIGHLOCK	High address space block lock These bits are used to lock the blocks of High Address Space from Program and Erase. The block numbering for the High Blocks starts at HIGHLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the HIGHLOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). When blocks are not present (due to configuration or total memory size), the HIGHLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 High Address Space Block is unlocked and can be modified. 1 High Address Space Block is locked and cannot be modified.

17.3.2.6 Lock 2 register (LOCK2)

The LOCK2 (256K Address Space Block Locking) register provides a means of protecting blocks from being modified. The A256KLOCK bits corresponding to nonexistent blocks are read as ‘1’.

To determine the effective status of the A256KLOCK Address Space, combine (Or) LOCK2 with sidebands:

- flh_plock2 – in order to evaluate protection against program
- flh_elock2 – in order to evaluate protection against erase

The effect of programming/erasing a block protected with LOCK2 is that no modification occurs and PEG is set to ‘1’.

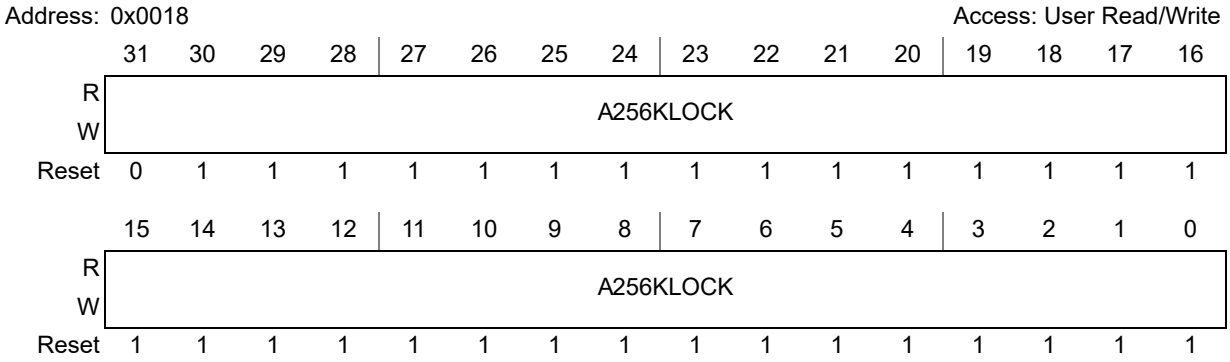


Figure 136. Lock 2 register (LOCK2)

Table 201. LOCK2 field descriptions

Field	Description
31:0 A256KLOCK	256K address space block lock A256KLOCK[31:0] These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the 256 KB blocks starts with A256KLOCK[0] and continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the A256KLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the A256KLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 A256K Address Space Block is unlocked and can be modified. 1 A256K Address Space Block is locked and cannot be modified.

17.3.2.7 Lock 3 register (LOCK3)

The LOCK3 (256K Address Space Block Locking) register provides a means of protecting blocks from being modified. The A256KLOCK bits corresponding to nonexistent blocks are read as ‘1’.

To determine the effective status of the A256KLOCK Address Space, combine (Or) LOCK3 with sidebands:

- flh_plock3 – in order to evaluate protection against program
- flh_elock3 – in order to evaluate protection against erase.

The effect of programming/erasing a block protected with LOCK3 is that no modification occurs and PEG is set to ‘1’.

Address: 0x001C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	A256KLOCK															
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	A256KLOCK															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 137. Lock 3 register (LOCK3)

Table 202. LOCK3 field descriptions

Field	Description
31:0 A256KLOCK	256K address space block lock A256KLOCK[63:32] These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the 256 KB blocks starts with A256KLOCK[32] and continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the A256KLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the A256KLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 A256K Address Space Block is unlocked and can be modified. 1 A256K Address Space Block is locked and cannot be modified.

17.3.2.8 Alternate Lock 0 register (LOCK0A)

The Alternate LOCK0A (Low/Mid Address Space Block Locking) register provides a means of protecting blocks from being modified on the alternate program and erase interface. LOWLOCKA and MIDLOCKA bits corresponding to non existent blocks are read as ‘1’.

The effective status of the Alternate Low Address Space is not affected by sidebands as in the Main Interface

The effect of programming/erasing a block protected with LOCK1A is that no modification occurs and PEGA is set to ‘1’.

Address: 0x0028 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TSLOCKA	0	LOWLOCKA													
W																
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MIDLOCKA															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 138. Alternate Lock 0 register (LOCK0A)

Table 203. LOCK0A field descriptions

Field	Description
31 TSLOCKA	Alternate UTEST NVM block lock This bit is used to lock the UTEST NVM from programming (UTEST NVM is not erasable). The LOCK field is not writable: – Once an Interlock Write is completed until MCRA.DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the TSLOCK bit is locked. The effective status of the UTEST protection is done combining (ORed) LOCK0A.TSLOCKA with sidebands *flh_aplock_utest* in order to evaluate protection against program. 0 The UTEST Block is unlocked and can be modified on the alternate interface. 1 The UTEST Block is locked and cannot be modified on the alternate interface.

Table 203. LOCK0A field descriptions (continued)

Field	Description
29:16 LOWLOCKA	Alternate Low address space block lock These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the Low Blocks starts at LOWLOCKA[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the LOWLOCKA bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the LOWLOCKA bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Alternate Low Address Space Block is unlocked and can be modified. 1 Alternate Low Address Space Block is locked and cannot be modified.
15:0 MIDLOCKA	Alternate Mid address space block lock These bits are used to lock the blocks of Mid Address Space from Program and Erase. The block numbering for the Mid Blocks starts at MIDLOCKA[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the MIDLOCKA bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the MIDLOCKA bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Alternate Mid Address Space Block is unlocked and can be modified. 1 Alternate Mid Address Space Block is locked and cannot be modified.

17.3.2.9 Alternate Lock 1 register (LOCK1A)

The Alternate LOCK1A (Alternate High Address Space Block Locking) register provides a means of protecting blocks from being modified on the alternate program and erase interface. HIGHLOCK bits corresponding to nonexistent blocks are read as ‘1’.

The effect of programming/erasing a block protected with LOCK1A is that no modification occurs and PEGa is set to ‘1’.

Address: 0x002C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HIGHLOCKA															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 139. Alternate Lock 1 register (LOCK1A)

Table 204. Alternate LOCK1A field descriptions

Field	Description
15:0 HIGHLOCKA	Alternate High address space block lock These bits are used to lock the blocks of High Address Space from Program and Erase. The block numbering for the High Blocks starts at HIGHLOCKA[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the HIGHLOCKA bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). When blocks are not present (due to configuration or total memory size), the HIGHLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 High Address Space Block is unlocked and can be modified. 1 High Address Space Block is locked and cannot be modified.

17.3.2.10 Select 0 register (SEL0)

The SEL0 (Low/Mid Address Space Block Select 0) register provides a means to select blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL0 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 17.4.3.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL0 at the start of the Erase operation) while writing has no effect.

Address: 0x0038 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	LOWSEL													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MIDSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 140. Select 0 register (SEL0)

Table 205. SEL0 field descriptions

Field	Description
29:16 LOWSEL	Low address space block Select The block numbering for the Low Blocks starts at LOWSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding LOWSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 Low Address Space Block is unselected for Erase. 1 Low Address Space Block is selected for Erase.

Table 205. SEL0 field descriptions (continued)

Field	Description
15:0 MIDSEL	Mid address space block Select The block numbering for the Mid Blocks starts at MIDLSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size), the corresponding MIDSEL bits default to unselected and are not writable.The reset value is always ‘0’, and register writes have no effect. 0 Mid Address Space Block is unselected for Erase. 1 Mid Address Space Block is selected for Erase.

17.3.2.11 Select 1 register (SEL1)

The SEL1 (High Address Space Block Select register 1) provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL1 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 17.4.3.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL1 at the start of the Erase operation) while writing has no effect.

Address: 0x003C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HIGHSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 141. Select 1 register (SEL1)

Table 206. SEL1 field descriptions

Field	Description
15:0 HIGHSEL	High/Data address space block Select 1 The block numbering for the High Blocks starts at HIGHSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding HIGHSEL bits default to unselected and are not writable. The reset value is always '0' and register writes have no effect. 0 High Address Space Block is unselected for Erase. 1 High Address Space Block is selected for Erase.

17.3.2.12 Select 2 register (SEL2)

The SEL2 (256K Address Space Block Select 2) register provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL2 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 17.4.3.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL2 at the start of the Erase operation) while writing has no effect.

Address: 0x0040 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 142. Select 2 register (SEL2)

Table 207. SEL2 field descriptions

Field	Description
31:0 A256KSEL	256K address space block select A256KSEL[31:0] The block numbering for the 256 KB blocks starts with A256KSEL[0] and continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding A256KSEL bits default to unselected and are not writable. The reset value is always '0', and register writes have no effect. 0 A256K Address Space Block is unselected for Erase. 1 A256K Address Space Block is selected for Erase.

17.3.2.13 Select 3 register (SEL3)

The SEL3 (256K Address Space Block Select 3) register provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL3 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 17.4.3.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL3 at the start of the Erase operation) while writing has no effect.

Address: 0x0044 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 143. Select 3 register (SEL3)

Table 208. SEL3 field descriptions

Field	Description
31:0 A256KSEL	256K address space block select A256KSEL[63:32] The block numbering for the 256 KB blocks starts with A256KSEL[32] and continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding A256KSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 A256K Address Space Block is unselected for Erase. 1 A256K Address Space Block is selected for Erase.

17.3.2.14 Module Configuration register 2 (MCR2)

The Parallel Module Configuration register is used to monitor all the Read operations of the flash memory module on Read-While-Read Partition 1.

Address: 0x0048 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	PAEE	PEEE	PDWEE				0	0	0	PSBC1	0	0	0	0
W			w1c	w1c								w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PEER	PRWE	PSBC	0	0	0	0	0	0	0	0	0	0	0	0	0
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 144. Module Configuration register 2 (MCR2)

Table 209. MCR2 field descriptions

Field	Description
29 PAEE	Parallel Address Encode Error On every Read request to the Flash, the incoming address is compared to an encoded address (row, column, and block) from the memory array using the read data sense amp timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error is recorded. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally 1 An Address Encode Error occurred during a previous Read.
28 PEEE	Parallel EDC after ECC Error PEEE provides information on previous reads monitoring the EDC after ECC feature. On every Read request to the Flash, ECC parity is recomputed (encoding) just after ECC decoding, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error is reported. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally 1 A previous Read may have been corrupted.
27:24 PDWEE	Parallel Double Word Uncorrectable ECC Error PDWEE identifies which Double Word is involved in ECC Uncorrectable Error Event. In Code Flash 4 Double Words are read in parallel and ECC is evaluated on a couple of Double Words basis. Therefore PDWEE value could be: 1111 if an ECC Uncorrectable Error is found in both couples of Double Words 1100 if an ECC Uncorrectable Error is found in the couple of Double Words at high address 0011 if an ECC Uncorrectable Error is found in the couple of Double Words at low address 0000 if no ECC Uncorrectable Error is found PDWEE is a read only. Flags bits get cleared when EER is cleared.
20 PSBC1	Parallel Single Bit Correction 1 PSBC1 provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection or ECC Double Error correction, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE1). 0 Reads are occurring normally. 1 An ECC Single Error occurred and was corrected during a previous Read. In Code Flash, with 128b ECC coding, ECC correction is evaluated on couple of Double word basis. In Data Flash for EEE, ECC correction is evaluated on each Double word.

Table 209. MCR2 field descriptions (continued)

Field	Description
15 PEER	Parallel ECC Event Error PEER provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. PEER may not be set to ‘1’ by the user. PEER is not set for a ECC correctable Error detection and related correction event. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 An ECC uncorrectable Error occurred during a previous Read. In Code Flash, with 128b ECC coding, ECC error is evaluated on couple of Double word basis. In Data Flash for EEE, ECC error is evaluated on each Double word.
14 PRWE	Parallel Read-while-Write event Error PRWE provides information on previous Reads during a Modify operation. Read-While-Write Error means that a Read access to the Flash Array partition has occurred while the FPEC was performing a Program or Erase operation or an Array Integrity Check on the same partition. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A RWW Error occurred during a previous Read.
13 PSBC	Parallel double Bit Correction PSBC provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE). 0 Reads are occurring normally. 1 An ECC Double Error occurred and was corrected during a previous Read. In Data Flash for EEE, ECC correction is evaluated on each Double word.

17.3.2.15 Program Address register (PAR)

The Program Address register provides the last successful address programmed. A successful program is defined as a program operation of data that is not all 1s, to an unlocked block and the operation results in MCR.PEG returning a ‘1’. This value is held until the next successful address program event.

Address: 0x004C Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	PAR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PAR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 145. Program Address register (PAR)

Table 210. PAR field descriptions

Field	Description
23:3 PAR	Address The Address register provides the last successful address programmed through the Main Interface. Address given is in Flash format and has to be intended as in the flash memory Map: offset respect to absolute address. Mapping can be found in *Section 17.3.2.16: Address register (ADR)*

17.3.2.16 Address register (ADR)

The Address register provides the first failing address in the event of module failures (ECC, RWW or FPEC). ADR captures the first failing address for the modes listed in *Table 212* according to the priority rules in *Table 213*.

In case of Array Integrity Check or Margin Read with Break Point enabled (UT0.AIBPE=1), the ADR value matches the address where the Break Point was triggered only when the current run (first start or re-start after a breakpoint (NAIBP cleared)) starts with MCR.ERR and MCR.SBC flags cleared, otherwise the ADR priority rule overrides Break Point.

Address: 0x0050 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SAD	0	0	0	0	0	0	0	ADDR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 146. Address register (ADR)

Table 211. ADR field descriptions

Field	Description
31 SAD	Test Address When this bit is high, the address indicated by ADDR[23:3] belongs to the Test Block.
23:3 ADDR	Address The Address register provides the first failing address in the event of ECC Uncorrectable error (MCR.EER set) or the first failing address in the event of RWW error (MCR.RWE set), or the address of a failure that may have occurred in a FPEC operation (MCR.PEG cleared). The Address register also provides the first address at which a ECC correction occurs (MCR.SBC or MCR.SBC1, or both are set) if the SoC is configured to show this feature through UT0.SBCE or UT0.SBCE1, or both. The ECC uncorrectable error detection takes the highest priority, followed by the RWW error, the FPEC error and the ECC error corrections. When accessed, ADR provides the address related to the first event occurred with the highest priority. In case of a simultaneous ECC Events Detection, bits ADDR[4] and ADDR[3] output ‘0’. In User mode the Address register is read only. In Modify Mode ADDR can be updated on FPEC Error on the whole flash memory address space. In User Test Mode ADDR can be updated on Error events on the whole flash memory address space. In Read and RWW Modes ADDR can be updated on Error belonging to the address space of the primary Read-While-Read Partition 0. Address given is in Flash format and has to be intended as in the flash memory Map.

Table 212. ADR update mode list

Mode	Update on error	Note
Read	ECC Uncorrectable Error Detection, ECC Double Error Correction (if UT0.SBCE=1) ECC Single Error Correction (if UT0.SBCE1=1)	—
Modify	FPEC error	—
RWW	Read-While-Write error	In RWW all Read and Modify cause of update on error may also occur
User Test	ECC Uncorrectable Error Detection, ECC Single Error Correction (if UT0.SBCE1=1)	With Break Point enabled (UT0.AIBPE=1), ADR requires MCR.ERR and MCR.SBC to be cleared

Table 213. ADR content: priority list

Priority level	Error flag	ADR content
1	MCR.EER = 1	Address of first ECC Uncorrectable Error
2	MCR.RWE = 1	Address of first RWW Error
3	MCR.PEG = 0	Address of first FPEC Error

Table 213. ADR content: priority list (continued)

Priority level	Error flag	ADR content
4	MCR.SBC = 1	Address of first ECC Double Error Correction
5	MCR.SBC1 = 1	Address of first ECC Single Error Correction

Table 214. Flash memory partition 0 address mapping

Block	Flash format addresses	Absolute addresses
B0F0	0x0088_0000 to 0x0088_7FFF	0x1800_0000 to 0x1800_7FFF
B0F1	0x0088_8000 to 0x0089_7FFF	0x1800_8000 to 0x1801_7FFF
B0F2	0x0089_8000 to 0x008A_7FFF	0x1801_8000 to 0x1802_7FFF
B0F3-UT	0x0080_0000 to 0x0080_3FFF	0x1FF8_0000 to 0x1FF8_3FFF
B0F3-BF	0x0080_4000 to 0x0080_7FFF	0x1FF0_0000 to 0x1FF0_3FFF
B0F3-TF	0x0080_8000 to 0x0080_BFFF	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—

Table 215. Flash memory partition 1 address mapping

Block	Flash format addresses	Absolute addresses
B1F0	0x0000_0000 to 0x0000_FFFF	0x0800_0000 to 0x0800_FFFF
B1F1	0x0001_0000 to 0x0001_FFFF	0x0801_0000 to 0x0801_FFFF
B1F2	0x0002_0000 to 0x0002_FFFF	0x0802_0000 to 0x0802_FFFF
B1F3	0x0003_0000 to 0x0006_FFFF	0x0803_0000 to 0x0806_FFFF
B1F4	0x0007_0000 to 0x000A_FFFF	0x0807_0000 to 0x080A_FFFF
B1F5	0x000B_0000 to 0x000E_FFFF	0x080B_0000 to 0x080E_FFFF
—	—	—
—	—	—
—	—	—

Table 215. Flash memory partition 1 address mapping (continued)

Block	Flash format addresses	Absolute addresses
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—

Table 216. Flash memory partition 2 address mapping

Block	Flash format addresses	Absolute addresses
B2F0	0x000F_0000 to 0x000F_FFFF	0x080F_0000 to 0x080F_FFFF
B2F1	0x0010_0000 to 0x0010_FFFF	0x0810_0000 to 0x0810_FFFF
B2F2	0x0011_0000 to 0x0011_FFFF	0x0811_0000 to 0x0811_FFFF
B2F3	0x0012_0000 to 0x0015_FFFF	0x0812_0000 to 0x0815_FFFF
B2F4	0x0016_0000 to 0x0016_FFFF	0x0816_0000 to 0x0819_FFFF
B2F5	0x001A_0000 to 0x001D_FFFF	0x081A_0000 to 0x081D_FFFF
—	—	—
—	—	—

Table 217. Flash memory EEPROM data partition 3 address mapping

Block	Flash format addresses	Absolute addresses
D0_00	0x0090_0000 to 0x0090_3FFF	0x08F0_0000 to 0x08F0_3FFF
D0_01	0x0090_4000 to 0x0090_7FFF	0x08F0_4000 to 0x08F0_7FFF
D0_02	0x0090_8000 to 0x0090_BFFF	0x08F0_8000 to 0x08F0_BFFF
D0_03	0x0090_C000 to 0x0090_FFFF	0x08F0_C000 to 0x08F0_FFFF

Table 218. Flash memory HSM EEPROM Data Partition 3 address mapping

Block	Flash format addresses	Absolute addresses
D0_04	0x0098_0000 to 0x0098_3FFF	0x18F0_0000 to 0x18F0_3FFF
D0_05	0x0098_4000 to 0x0098_7FFF	0x18F0_4000 to 0x18F0_7FFF

17.3.2.17 User Test 0 register (UT0)

The User Test 0 register feature gives the user of the flash memory module the ability to perform test features on the Flash.

The User Test 0 register allows control of the way the Flash content check is performed. Bits MRE, MRV and AIS of the User Test 0 register are not accessible when MCR.DONE or UT0.AID are low. Reading returns indeterminate data while writing has no effect.

Address: 0x0054 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UTE	SBCE	0	0	0	0	0	0	SBCE1	0	0	0	DDEF	CPR	CPA	CPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	VTD	0	0	NAIBP	AIBPE	0	AISUS	MRE	MRV	0	AIS	AIE	AID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 147. User Test 0 register (UT0)

Table 219. UT0 field descriptions

Field	Description
31 UTE	User Test Enable This status bit gives indication when User Test is enabled. All bits in UT0 and UM0–UM9 are locked when this bit is ‘0’. This bit is not writable to a ‘1’, but may be cleared. The reset value is ‘0’. To set UTE, write the password 0xF9F99999 to the UT0 register. The UTE password is only accepted if PGM=0 and ERS=0 (Program and Erase are not being requested). UTE can only be cleared if AID=1, MRE=0 and AIE=0. While clearing UTE, writes to set AIE or MRE are ignored.
30 SBCE	Double Bit Correction Enable This bit, when high, enables to flag the information about any eventual ECC Double Bit Correction in the Flash array (MCR.SBC and ADR). This is true in read mode and in Array Integrity Check. SBCE is also used as an Enable for interrupt signals created by the embedded flash memory. ECC corrections that occur when SBCE is cleared are not logged.
23 SBCE1	Single Bit Correction Enable This bit, when high, enables to flag the information about any eventual ECC Single Bit Correction in the Flash array (MCR.SBC1 and ADR). This is true in read mode and in Array Integrity Check. SBCE1 is also used as an Enable for interrupt signals created by the embedded flash memory. ECC corrections that occur when SBCE1 is cleared are not logged.
19 DDEF	Disable Data memory Edc post ecc error reporting to FCCU This bit is used to disable the EEE error reporting when reading Data Memory. Code Memory Edc post Ecc error reporting is not affected. MCR.EEE is generated regarless of DDEF. 0 Data memory EEE error reporting to FCCU is enabled. 1 Data memory EEE error reporting to FCCU is disabled.

Table 219. UT0 field descriptions (continued)

Field	Description
18 CPR	Customer Programmable Read Voltage and Reference Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block when reading Read-While-Read partition 0. This bit is used to control the error generation for the Customer Programmable Detection area in the RWR1_UTEST block when reading Read-While-Read partition 1. If this bit is set and the Customer Programmable Detection location in UTEST is read, a Read Voltage and Reference Error detection is set. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled.
17 CPA	Customer Programmable Address Encode Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block when reading Read-While-Read partition 0. This bit is used to control the error generation for the Customer Programmable Detection area in the RWR1_UTEST block when reading Read-While-Read partition 1. If this bit is set, and the Customer Programmable Detection location in UTEST is read an Address Encode Error detection is set. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
16 CPE	Customer Programmable EDC after ECC Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block when reading Read-While-Read partition 0. This bit is used to control the error generation for the Customer Programmable Detection area in the RWR1_UTEST block when reading Read-While-Read partition 1. If this bit is set, and the Customer Programmable Detection location in UTEST is read a EDC after ECC Error detection is set. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
12 VTD	Vth Distribution To enable Vth Distribution, VTD may be set. Effective setting of this bit is allowed when SoC has been brought into Failure Analysis Mode. When Vth Distribution is enabled, multiple run are needed to assess the Vth status of the cells belonging to selected blocks. At the end of each run NAIPB has to be checked and if equals to ‘1’ it has to be cleared to start an increased voltage step run. See NAIBP description for more information. 0 Vth Distribution is not enabled. 1 Vth Distribution is enabled.

Table 219. UT0 field descriptions (continued)

Field	Description
9 NAIBP	Next Array Integrity Break Point If AIBPE is set, NAIBP is set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to ‘1’ but may be cleared to ‘0’. Clearing NAIBP to ‘0’ enables the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID is set with NAIBP remaining ‘0’. If VTD is set, NAIBP is set once a voltage step run has been completed during the voltage sweep in Vth Distribution. NAIBP is not writable to ‘1’ but may be cleared to ‘0’. Clearing NAIBP to ‘0’ enables the Vth Distribution run to be re-started with an increased voltage step. If the Vth Distribution completes the last voltage step run, AID is set with NAIBP remaining ‘0’. 0 Array Integrity-Vth Distribution is not currently at a break point. 1 Array Integrity-Vth Distribution is at a break point.
8 AIBPE	Array Integrity Break Point Enable To enable breakpoints during an Array Integrity test, AIBPE may be set. When breakpoints are enabled, if the run has been started with MCR.ERR (and MCR.SBC when UT0.SBCE=1) flag cleared, on the eventual occurrence of a breakpoint condition, MCR flags and ADR value may be updated. See NAIBP description for more information about Array Integrity breakpoints. 0 Array Integrity breakpoints are not enabled. 1 Array Integrity breakpoints are enabled during Array Integrity Checks.
6 AISUS	Array Integrity Suspend AISUS toggles Suspend of an Array Integrity - Margin Read Operation. AISUS can be written to ‘1’ only when AID is low AISUS can be written to ‘0’ only when AID is high. Attempting to Write AISUS and AIE on the same clock cycle results in only AIE being written. 0 Array Integrity sequence not suspended. 1 Array Integrity sequence is suspended.
5 MRE	Margin Read Enable MRE enables margin reads to be done. MRE combined with MRV enables User Margin mode reads to be done. Margin reads run on a block-by-block basis in linear address sequence selected by SEL0, SEL1 and SEL2. This bit is not accessible when MCR.DONE or UT0.AID are low or if AISUS or NAIBP are high: reading returns indeterminate data while writing has no effect. 0 Margin Read is not enabled. 1 Margin Read is enabled.
4 MRV	Margin Read Value If MRE is high, MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). This bit is not accessible whenever MCR.DONE or UT0.AID are low or if AISUS or NAIBP are high: reading returns indeterminate data while writing has no effect. 0 Zero’s (programmed) margin reads are requested (if MRE=1). 1 One’s (erased) margin reads are requested (if MRE=1).

Table 219. UT0 field descriptions (continued)

Field	Description
2 AIS	Array Integrity Sequence AIS determines the address sequence to be used during Array Integrity checks. AIS is not writable if AIE is high. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 1 Array Integrity sequence must be set to one for sequential address.
1 AIE	Array Integrity Enable Setting AIE to '1' starts the Array Integrity Check and the Margin Read on all selected blocks. The sequence is selected by AIS and the MISR (UM0–UM9) is checked on completion to determine if a correct signature is obtained. AIE can be set only if MCR.ERS, MCR.PGM and MCR.EHV are all low. AIE is not simultaneously writable to '1' while UTE is being cleared to '0'. Clearing AIE terminates the check when an Array Integrity operation finishes (AID=1) and aborts the check when and Array Integrity operation is ongoing (AID=0). AIE remains '1' when the Array Integrity Check - Margin read are suspended or are in breakpoint and cannot be cleared. (AIE can be cleared if NAIBP=0 and AISU=0) 0 Array Integrity Check and Margin Read and are not enabled. 1 Array Integrity Check and Margin Read are enabled.
0 AID	Array Integrity Done AID is set to indicate that the Array Integrity Check - Margin Read is complete and the MISR (UM0–9) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set) and the breakpoint condition is met. AID may also assert if Suspend is set (AISUS 0–1). AID can not be written, and is status only. 0 Array Integrity Check is on-going. 1 Array Integrity Check is done.

17.3.2.18 User Multiple Input Signature registers

The Multiple Input Signature registers allow evaluation of the Array Integrity or Margin Read run on a block-by-block basis by accumulating signatures from data bit fields.

These registers are not accessible whenever MCR.DONE or UT0.AID are low. Reading returns indeterminate data while writing has no effect.

The registers can be written in order to seed signatures before starting an AIC or MR operation.

Note: Do not write to the MISR registers during an Array Integrity operation (including Suspend or Breakpoint) as it may alter the final signature in an unpredictable way.

If reads extend into an invalid address location during address sequencing (that is greater than the maximum address for a given array size), reads are still executed to the array but the results are not deterministic: MISR registers are not recalculated and the previous value is retained.

17.3.2.18.1 User Multiple Input Signature 0 register (UM0)

UM0 represents bits 31–0 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0058 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR31	MISR30	MISR29	MISR28	MISR27	MISR26	MISR25	MISR24	MISR23	MISR22	MISR21	MISR20	MISR19	MISR18	MISR17	MISR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR15	MISR14	MISR13	MISR12	MISR11	MISR10	MISR9	MISR8	MISR7	MISR6	MISR5	MISR4	MISR3	MISR2	MISR1	MISR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 148. User Multiple Input Signature 0 register (UM0)

Table 220. UM0 field descriptions

Field	Description
31:0 MISR[31:0]	Multiple input signature 31–0 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.18.2 User Multiple Input Signature 1 register (UM1)

UM1 represents bits 63–32 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x005C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR63	MISR62	MISR61	MISR60	MISR59	MISR58	MISR57	MISR56	MISR55	MISR54	MISR53	MISR52	MISR51	MISR50	MISR49	MISR48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR47	MISR46	MISR45	MISR44	MISR43	MISR42	MISR41	MISR40	MISR39	MISR38	MISR37	MISR36	MISR35	MISR34	MISR33	MISR32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 149. User Multiple Input Signature 1 register (UM1)

Table 221. UM1 field descriptions

Field	Description
31:0 MISR[63:32]	Multiple input signature 63–32 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.18.3 User Multiple Input Signature 2 register (UM2)

UM2 represents bits 95–64 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0060 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR95	MISR94	MISR93	MISR92	MISR91	MISR90	MISR89	MISR88	MISR87	MISR86	MISR85	MISR84	MISR83	MISR82	MISR81	MISR80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR79	MISR78	MISR77	MISR76	MISR75	MISR74	MISR73	MISR72	MISR71	MISR70	MISR69	MISR68	MISR67	MISR66	MISR65	MISR64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 150. User Multiple Input Signature 2 register (UM2)

Table 222. UM2 field descriptions

Field	Description
31:0 MISR[95:64]	Multiple input signature 95–64 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.18.4 User Multiple Input Signature 3 register (UM3)

UM3 represents bits 127–96 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0064 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR127	MISR126	MISR125	MISR124	MISR123	MISR122	MISR121	MISR120	MISR119	MISR118	MISR117	MISR116	MISR115	MISR114	MISR113	MISR112
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR111	MISR110	MISR109	MISR108	MISR107	MISR106	MISR105	MISR104	MISR103	MISR102	MISR101	MISR100	MISR99	MISR98	MISR97	MISR96
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 151. User Multiple Input Signature 3 register (UM3)

Table 223. UM3 field descriptions

Field	Description
31:0 MISR[127:96]	Multiple input signature 127–96 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.18.5 User Multiple Input Signature 4 register (UM4)

UM4 represents bits 159–128 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0068 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR159	MISR158	MISR157	MISR156	MISR155	MISR154	MISR153	MISR152	MISR151	MISR150	MISR149	MISR148	MISR147	MISR146	MISR145	MISR144
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR143	MISR142	MISR141	MISR140	MISR139	MISR138	MISR137	MISR136	MISR135	MISR134	MISR133	MISR132	MISR131	MISR130	MISR129	MISR128
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 152. User Multiple Input Signature 4 register (UM4)

Table 224. UM4 field descriptions

Field	Description
31:0 MISR[159:128]	Multiple input signature 159–128 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.18.6 User Multiple Input Signature 5 register (UM5)

UM5 represents bits 191–160 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x006C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR191	MISR190	MISR189	MISR188	MISR187	MISR186	MISR185	MISR184	MISR183	MISR182	MISR181	MISR180	MISR179	MISR178	MISR177	MISR176
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR175	MISR174	MISR173	MISR172	MISR171	MISR170	MISR169	MISR168	MISR167	MISR166	MISR165	MISR164	MISR163	MISR162	MISR161	MISR160
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 153. User Multiple Input Signature 5 register (UM5)

Table 225. UM5 field descriptions

Field	Description
31:0 MISR[191:160]	Multiple input signature 191–160 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.18.7 User Multiple Input Signature 6 register (UM6)

UM6 represents bits 223–192 of the whole 288 bits word (4 doublewords including ECC).

Address: 0x0070 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR223	MISR222	MISR221	MISR220	MISR219	MISR218	MISR217	MISR216	MISR215	MISR214	MISR213	MISR212	MISR211	MISR210	MISR209	MISR208
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR207	MISR206	MISR205	MISR204	MISR203	MISR202	MISR201	MISR200	MISR199	MISR198	MISR197	MISR196	MISR195	MISR194	MISR193	MISR192
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 154. User Multiple Input Signature 6 register (UM6)

Table 226. UM6 field descriptions

Field	Description
31:0 MISR[223:192]	Multiple input signature 223–192 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.18.8 User Multiple Input Signature 7 register (UM7)

UM7 represents bits 255–224 of the whole 288 bits word (4 doublewords including ECC).

Address: 0x0074 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR255	MISR254	MISR253	MISR252	MISR251	MISR250	MISR249	MISR248	MISR247	MISR246	MISR245	MISR244	MISR243	MISR242	MISR241	MISR240
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR239	MISR238	MISR237	MISR236	MISR235	MISR234	MISR233	MISR232	MISR231	MISR230	MISR229	MISR228	MISR227	MISR226	MISR225	MISR224
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 155. User Multiple Input Signature 7 register (UM7)

Table 227. UM7 field descriptions

Field	Description
31:0 MISR[255:224]	Multiple input signature 255–224 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.18.9 User Multiple Input Signature 8 register (UM8)

UM8 represents the ECC parity bits of the whole 288-bits word (8 parity bits for each of the 4 doublewords).

Address: 0x0078 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR287	MISR286	MISR285	MISR284	MISR283	MISR282	MISR281	MISR280	MISR279	MISR278	MISR277	MISR276	MISR275	MISR274	MISR273	MISR272
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR271	MISR270	MISR269	MISR268	MISR267	MISR266	MISR265	MISR264	MISR263	MISR262	MISR261	MISR260	MISR259	MISR258	MISR257	MISR256
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 156. User Multiple Input Signature 8 register (UM8)

Table 228. UM8 field descriptions

Field	Description
31:0 MISR[287:256]	Multiple input signature 287–256 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.18.10 User Multiple Input Signature 9 register (UM9)

UM9 represents the ERR bits.

Address: 0x007C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MISR319	MISR318	MISR317	MISR316	MISR315	MISR314	MISR313	MISR312	MISR311	MISR310	MISR309	MISR308	MISR307	MISR306	MISR305	MISR304
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MISR303	MISR302	MISR301	MISR300	MISR299	MISR298	MISR297	MISR296	MISR295	MISR294	MISR293	MISR292	MISR291	MISR290	MISR289	MISR288
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 157. User Multiple Input Signature 9 register (UM9)

Table 229. UM9 field descriptions

Field	Description
31:0 MISR[319:288]	Multiple input signature 319–288 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

17.3.2.19 Over-program protection registers

Over-program protection prevents blocks from being over-programmed. These registers show the over-program protection status. See *Section 17.4.3.1.2: Over-programming protection (OPP) enable*.

17.3.2.19.1 Over-program protection 0 register (OPP0)

The Low/Mid Address Space is covered by the Over Program Protection 0 (OPP0) register.

Address: 0x0080 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	LOWOPP													
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MIDOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 158. Over-program protection 0 register (OPP0)

Table 230. OPP0 field descriptions

Field	Description
29:16 LOWOPP	Low address space Over Program Protection The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWOPP field is not writable, and is status only. The default value of the LOWOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the LOWOPP bits default to not over-programming protected and will remain 0. 0 Low Address Space Block is unprotected from over-programming. 1 Low Address Space Block is protected from over-programming.
15:0 MIDOPP	Mid address space Over Program Protection The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDOPP field is not writable, and is status only. The default value of the MIDOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the MIDOPP bits default to not over-programming protected, and will remain 0. 0 Mid Address Space Block is unprotected from over-programming. 1 Mid Address Space Block is protected from over-programming.

17.3.2.19.2 Over-program protection 1 register (OPP1)

The High Address Space is covered by the Over Program Protection 1 (OPP1) register.

Address: 0x0084 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HIGHOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 159. Over-program protection 1 register (OPP1)

Table 231. OPP1 field descriptions

Field	Description
15:0 HIGHOPP	High address space Over Program Protection The block numbering for the High Blocks starts at HIGHOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHOPP field is not writable, and is status only. The default value of the HIGHOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the HIGHOPP bits default to not over-programming protected and will remain 0. 0 High Address Space Block is unprotected from over-programming. 1 High Address Space Block is protected from over-programming.

17.3.2.19.3 Over-program protection 2 register (OPP2)

The A256K Address Space Block is covered by the Over Program Protection 2 (OPP2) register.

Address: 0x0088 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 160. Over-program protection 2 register (OPP2)

Table 232. OPP2 field descriptions

Field	Description
31:0 A256KOPP	256K Address Space Block Over Program Protection A256KOPP[31:0] The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted for. The A256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the A256KOPP bits default to not over-programming protected and will remain 0. 0 A256K Address Space Block is unprotected from over-programming. 1 A256K Address Space Block is protected from over-programming.

17.3.2.19.4 Over-program protection 3 register (OPP3)

The 256K high Address Space is covered by the Over Program Protection 3 (OPP3) register.

Address: 0x008C Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 161. Over-program protection 3 register (OPP3)

Table 233. OPP3 field descriptions

Field	Description
31:0 A256KOPP	256K Address space Over Program Protection A256KOPP[63:32] The block numbering for the 256 KB Blocks starts at A256KOPP[32] and continues until all blocks are accounted for. The A256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the A256KOPP bits default to not over-programming protected and will remain 0. 0 A256K Address Space Block is unprotected from over-programming. 1 A256K Address Space Block is protected from over-programming.

17.3.2.20 Test Mode Disable Password Check register (TMD)

The Test Mode Disable Password Check (TMD) register provides a means of entering the password to be matched with the one stored in UTEST location in order to allow manufacturer access to protected Test Mode blocks by writing 0x2D3C_4B5A in the Test Mode Disable Seal location.

Address: 0x0090 Access: User Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 162. Test Mode Disable Password Check register (TMD)

Table 234. TMD field descriptions

Field	Description
31:0 PWD	The Test Mode Disable Check register is used as a location to provide a password challenge to unlock blocks selected to be sealed as part of the Test Mode Disable seal mechanism. Please see *Section 17.4.6.2: Test Mode Disable* for more information. Writes to the register have no effect except for a password challenge. Reads to this register will always return 0. The register's reset value is 0. **Note:**

17.3.2.21 HSM Alternate selector registers

HSM Alternate selectors assign blocks from default to ALTERNATE Program & Erase interface. These registers show the HSM Alternate assignment status. See *Section 17.4.4: Alternate program and erase interface*.

17.3.2.21.1 HSM ALT-SEL register 0 (ALTSEL0)

The Low/Mid Address Space is assigned between MAIN and ALT Interface. Through the HSM ALT-SEL register it is possible to read the actual status of the assignment.

Address: 0x0094 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	LOWALTSEL													
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MIDALTSEL															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 163. HSM ALT-SEL register 0 (ALTSEL0)

Table 235. ALTSEL0 field descriptions

Field	Description
29:16 LOWALTSEL	Low address space HSM Alternate Select The block numbering for the Low Blocks starts at LOWALTSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWALTSEL field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the LOWALTSEL bits will remain 0. 0 Low Address Space Block is assigned to Main interface. 1 Low Address Space Block is assigned to Alternate interface.
15:0 MIDALTSEL	Mid address space HSM Alternate Select 0 The block numbering for the Mid Blocks starts at MIDALTSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDALTSEL field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the MIDALTSEL bits will remain 0. 0 Mid Address Space Block is assigned to Main interface. 1 Mid Address Space Block is assigned to Alternate interface.

17.3.2.21.2 HSM ALT-SEL register 1 (ALTSEL1)

The High Address Space is assigned between MAIN and ALT Interface. Through the HSM ALT-SEL register it is possible to read the actual status of the assignment.

Address: 0x0098 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HIGHALTSEL															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 164. HSM ALT-SEL register 1 (ALTSEL1)

Table 236. ALTSEL1 field descriptions

Field	Description
15:0 HIGHALTSEL	High address space HSM Alternate Select The block numbering for the High Blocks starts at HIGHALTSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHALTSEL field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the HIGHALTSEL bits will remain 0. 0 High Address Space Block is assigned to Main interface. 1 High Address Space Block is assigned to Alternate interface.

17.3.2.22 Erase Lock protection registers

Erase Lock protection prevents blocks from being erased. These registers show the Erase Lock protection status. See *Section 17.4.6.1: Modify protection*.

17.3.2.22.1 Erase Lock Protection 0 register (ELOCK0)

The Low/Mid Address Space is covered by the Erase Lock Protection 0 (ELOCK0) register.

Address: 0x0100 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOWELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MIDELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 165. Erase Lock Protection 0 register (ELOCK0)

Table 237. ELOCK0 field descriptions

Field	Description
31:16 LOWELOCK	Low address space Erase Lock Protection The block numbering for the Low Blocks starts at LOWELOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the LOWELOCK bits default is Erase Lock protected and will remain 1. 0 Low Address Space Block is Erase unprotected. 1 Low Address Space Block is Erase protected.

Table 237. ELOCK0 field descriptions (continued)

Field	Description
15:0 MIDELOCK	Mid address space Erase Lock Protection The block numbering for the Mid Blocks starts at MIDELOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the MIDELOCK bits default is Erase Lock protected and will remain 1. 0 Mid Address Space Block is Erase unprotected. 1 Mid Address Space Block is Erase protected.

17.3.2.22.2 Erase Lock Protection 1 register (ELOCK1)

The High Address Space is covered by the Erase Lock Protection 1 (ELOCK1) register.

Address: 0x0104 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HIGHELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

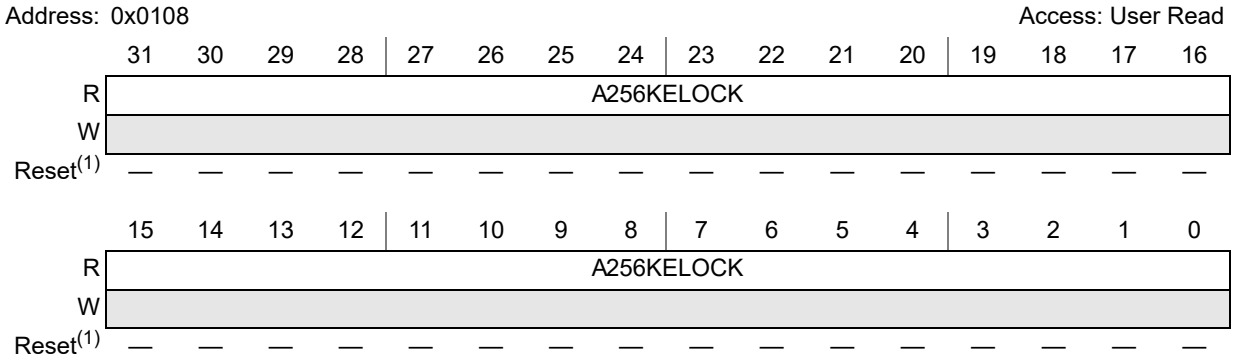
Figure 166. Erase Lock Protection 1 register (ELOCK1)

Table 238. ELOCK1 field descriptions

Field	Description
15:0 HIGHELOCK	High address space Erase Lock Protection The block numbering for the High Blocks starts at HIGHELOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the HIGHELOCK bits default is Erase Lock protected and will remain 1. 0 High Address Space Block is Erase unprotected. 1 High Address Space Block is Erase protected.

17.3.2.22.3 Erase Lock Protection 2 register (ELOCK2)

The 256K Address Space is covered by the Erase Lock Protection 2 (ELOCK2) register.



1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

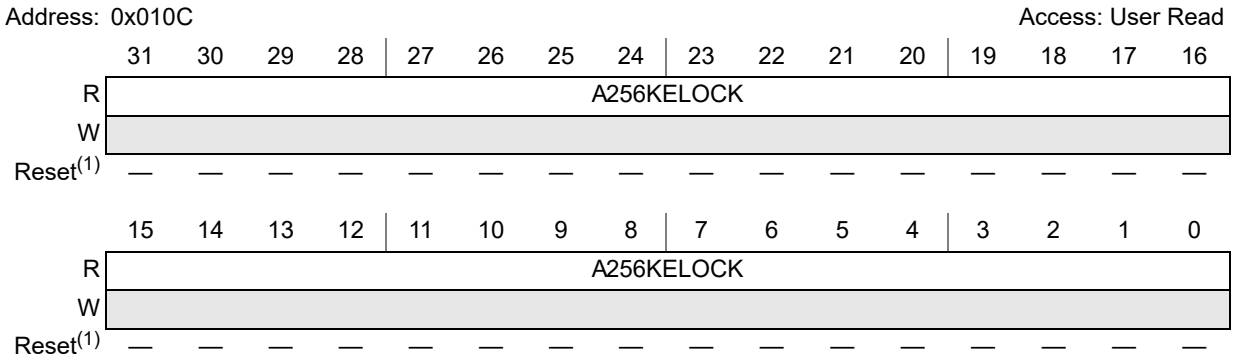
Figure 167. Erase Lock Protection 2 register (ELOCK2)

Table 239. ELOCK2 field descriptions

Field	Description
31:0 A256KELOCK	256K High address space Erase Lock Protection A256KELOCK[31:0] The block numbering for the 256 KB Blocks starts at A256KELOCK[0] and continues until all blocks are accounted for. The A256KELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KELOCK bits default is Erase Lock protected and will remain 1. 0 A256K Address Space Block is Erase unprotected. 1 A256K Address Space Block is Erase protected.

17.3.2.22.4 Erase Lock protection 3 register (ELOCK3)

The 256K Address Space is covered by the Erase Lock Protection 3 (ELOCK3) register.



1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 168. Erase Lock protection 3 register (ELOCK3)

Table 240. ELOCK3 field descriptions

Field	Description
31:0 A256KELOCK	256K High address space Erase Lock Protection A256KELOCK[63:32] The block numbering for the 256 KB Blocks starts at A256KELOCK[32] and continues until all blocks are accounted for. The A256KELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KELOCK bits default is Erase Lock protected and will remain 1. 0 A256K Address Space Block is erase unprotected. 1 A256K Address Space Block is Erase protected.

17.3.2.23 Program Lock protection registers

Program Lock protection prevents blocks from being programmed. These registers show the Program Lock protection status. See *Section 17.4.6.1: Modify protection*.

17.3.2.23.1 Program Lock Protection 0 register (PLOCK0)

The Low/Mid Address Space is covered by the Program Lock Protection 0 (PLOCK0) register.

Address: 0x0110 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOWPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MIDPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 169. Program Lock Protection 0 register (PLOCK0)

Table 241. PLOCK0 field descriptions

Field	Description
31:16 LOWPLOCK	Low address space Erase Lock Protection The block numbering for the Low Blocks starts at LOWPLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the LOWPLOCK bits default is Program Lock protected and will remain 1. 0 Low Address Space Block is Program unprotected. 1 Low Address Space Block is Program protected.

Table 241. PLOCK0 field descriptions (continued)

Field	Description
15:0 MIDPLOCK	Mid address space Erase Lock Protection The block numbering for the Mid Blocks starts at MIDPLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the MIDPLOCK bits default is Program Lock protected and will remain 1. 0 Mid Address Space Block is Program unprotected. 1 Mid Address Space Block is Program protected.

17.3.2.23.2 Program Lock Protection 1 register (PLOCK1)

The High Address Space is covered by the Program Lock Protection 1 (PLOCK1) register.

Address: 0x0114 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HIGHPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

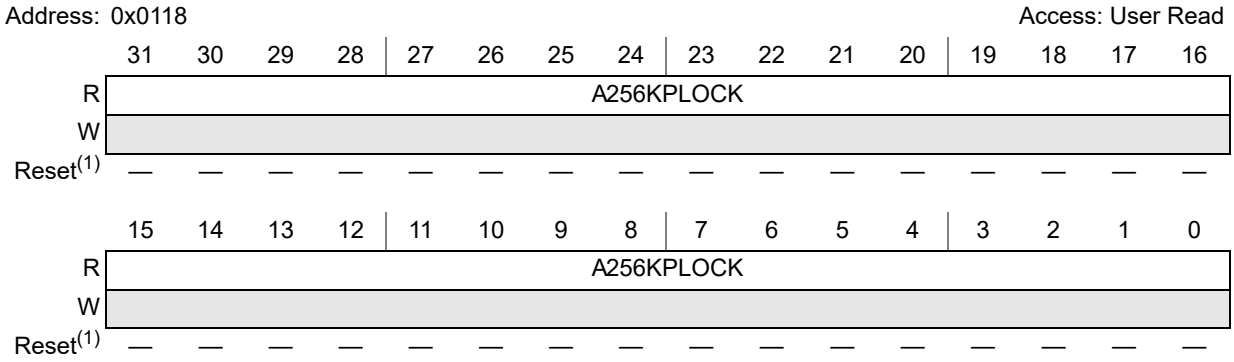
Figure 170. Erase Lock Protection 1 register (PLOCK1)

Table 242. PLOCK1 field descriptions

Field	Description
15:0 HIGHPLOCK	High address space Program Lock Protection The block numbering for the High Blocks starts at HIGHPLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the HIGHPLOCK bits default is Program Lock protected and will remain 1. 0 High Address Space Block is Program unprotected. 1 High Address Space Block is Program protected.

17.3.2.23.3 Program Lock Protection 2 register (PLOCK2)

The 256K Address Space is covered by the Program Lock Protection 2 (PLOCK2) register.



1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

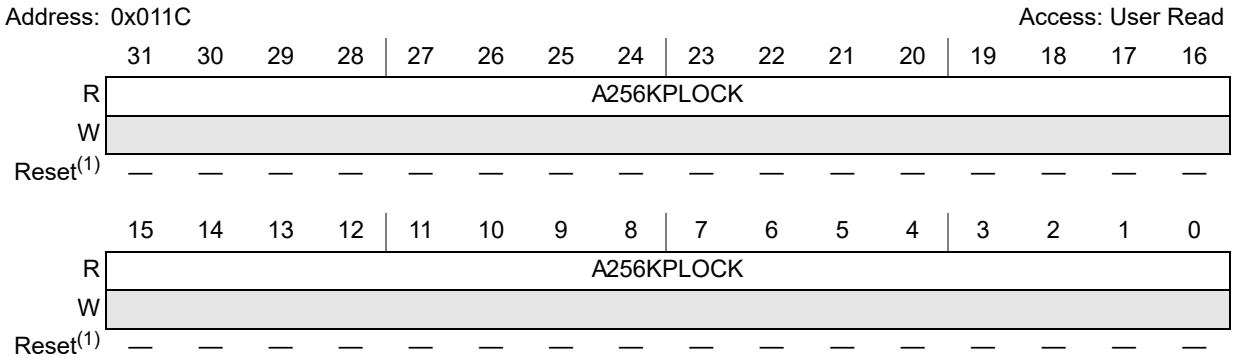
Figure 171. Program Lock Protection 2 register (PLOCK2)

Table 243. PLOCK2 field descriptions

Field	Description
31:0 A256KPLOCK	256K High address space Program Lock Protection A256KPLOCK[31:0] The block numbering for the 256 KB Blocks starts at A256KPLOCK[0] and continues until all blocks are accounted for. The A256KPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KPLOCK bits default is Program Lock protected and will remain 1. 0 A256K Address Space Block is Program unprotected. 1 A256K Address Space Block is Program protected.

17.3.2.23.4 Program Lock protection 3 register (PLOCK3)

The 256K Address Space is covered by the Program Lock Protection 3 (PLOCK3) register.



1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 172. Program Lock protection 3 register (PLOCK3)

Table 244. PLOCK3 field descriptions

Field	Description
31:0 A256KPLOCK	256K High address space Program Lock Protection A256KPLOCK[63:32] The block numbering for the 256 KB Blocks starts at A256KPLOCK[32] and continues until all blocks are accounted for. The A256KPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KPLOCK bits default is Program Lock protected and will remain 1. 0 A256K Address Space Block is Program unprotected. 1 A256K Address Space Block is Program protected.

17.3.2.24 Parallel Address register (ADR2)

The Parallel Address register provides the first failing address in the event of module failures (ECC) or the first address at which ECC single error correction occurs, when Reads are performed from Read While Read 1 interface. ADR2 captures the first failing address for the modes listed in *Table 246* according to the priority rules in *Table 247*.

Address: 0x0134 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	PADR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PADR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 173. Parallel Address register (ADR2)

Table 245. ADR2 field descriptions

Field	Description
23:3 PADR	Parallel Address The Address register provides the first failing address in the event of ECC Uncorrectable error (MCR2.PEER set) or the first failing address in the event of RWW error (MCR2.PRWE set). The Address register also provides the first address at which a ECC correction occurs (MCR2.PSBC or MCR2.PSBC1, or both are set) if the SoC is configured to show this feature through UT0.SBCE or UT0.SBCE1, or both. The ECC uncorrectable error detection takes the highest priority, followed by the RWW error, and the ECC error corrections. When accessed, PADR provides the address related to the first event occurred with the highest priority. In case of a simultaneous ECC Events Detection, bits PADR[4] and PADR[3] output '0'. In Read and RWW Modes PADR can be updated on Error belonging to the address space of the primary Read-While-Read Partition 1. Address given is in Flash format and has to be intended as in the flash memory Map.

Table 246. ADR2 update mode list

Mode	Update on error	Note
Read	ECC Uncorrectable Error Detection, ECC Single Error Correction (if UT0.SBCE1=1)	—
RWW	Read-While-Write error	In RWW all Read and Modify cause of update on error may also occur
User Test	ECC Uncorrectable Error Detection, ECC Single Error Correction (if UT0.SBCE1=1)	With Break Point enabled (UT0.AIBPE=1), ADR requires MCR.ERR and MCR.SBC to be cleared

Table 247. ADR2 content: priority list

Priority level	Error flag	ADR2 content
1	MCR2.PEER = 1	Address of first ECC Uncorrectable Error
2	MCR2.PRWE = 1	Address of first RWW Error
3	MCR2.PSBC1 = 1	Address of first ECC Single Error Correction

For the address mapping, refer to *Section 17.3.2.16: Address register (ADR)* description.

17.4 Functional description

This chapter describes the details of all the modes of operation available to the flash memory module.

17.4.1 Reset

A reset is the highest priority operation for the flash memory module and terminates all other operations.

The flash memory module uses reset to initialize register and status bits to their default reset values. If the flash memory module is executing a Program or Erase operation (MCR.PGM=1 or MCR.ERS=1) and a reset is issued, the operation is immediately terminated and the module disables the high voltage circuits.Reset terminates all operations and forces the flash memory module into Read mode ready to receive Read/Write accesses.

Warning: Reset and Power-Off must not be used as a systematic way to terminate a Program or Erase operation.

After reset is negated Read register access may be performed, although registers that require updating from shadow information or other inputs may not read updated values until MCR.DONE transforms. MCR.DONE may be polled to determine if the flash memory module has transitioned out of reset. Note that the registers cannot be written until MCR.DONE is high.

17.4.2 Read mode

Read mode is the active mode of operation of the flash memory module in which the Flash array can be read or ‘written’ (the so-called Interlock Write), or the Flash registers can be read or written.

The flash memory module enters Read mode when exiting from reset. The default state of the flash memory module is Read mode.

The main spaces (Low, Mid, High, and 256K) and UTEST address space of Flash array can only be read in Read mode.

Flash memory module is in Read mode, granting the same access time, under the following conditions:

- The Read state is active when the module is enabled (User Mode Read)
- The Read state is active when MCR.ERS and MCR.ESUS are high and MCR.PGM is low (Erase Suspend)
- The Read state is active when MCR.PGM and MCR.PSUS are high (Program Suspend)
- The Read state is active when MCR.PGM or MCR.ERS are high and High Voltage operation is ongoing on one partition, thus MCR.EHV is high (condition referred to as Read-While-Write). In this case reads done on the other partitions are allowed, while reads done to the partition being operated on (either Erase or Program) result in an error with MCR.RWE set and the data read back possibly being random including the possibility of ECC errors.
- The Read state is active when MCRA.ERSA and MCRA.ESUSA are high and MCRA.PGMA is low (Alternate Erase Suspend)
- The Read state is active when MCRA.PGMA and MCRA.PSUSA are high (Alternate Program Suspend)
- The Read state is active when MCRA.PGMA or MCRA.ERSA are high and High Voltage operation is ongoing on one partition, thus MCRA.EHVa is high (condition referred to as Alternate Read-While-Write). In this case reads done on the other partitions are allowed, while reads done to the partition being operated on (either Alternate Erase or Alternate Program) result in an error with MCRA.RWEA set and the data read back possibly being random including the possibility of ECC errors.

17.4.2.1 Read While Read

Flash memory module is organized in Two independent Read interfaces. Each Read interface is associated exclusively to a specific group of Read While Write Partitions. flash memory module supports independent and Parallel Read access through the Two interfaces, namely Read-While-Read. Read interface assignment to Read While Write Partition is described in the following table:

Table 248. Read interface assignment

Read-While-Read Interface 0	Read-While Read Interface 1
RWW P0	—
RWW P1	—
—	RWW P2
RWW P3	—

Code Flash partitions reads return 256 bits (1 page = 4 doublewords). Data Flash partition reads return 128 bits. Register reads return 32 bits (1 word).

Flash array reads are performed through the Bus Interface Unit. In many cases the BIU applies “Read Page Buffering” to allow sequential reads to be executed with higher performance. This can provide data coherency issues that must be handled by software. Data coherency may be an issue after a program or an Erase operation, as well as Test block operations.

Flash array reads to invalid locations result in indeterminate data. Invalid locations occur when addressing points to blocks that are not included in the flash memory map of any partition. Interlock writes to invalid locations result in an interlock occurring, but attempts to program these blocks do not eventuate since they are forced into a locked state. Erase occurs to selected and unlocked blocks even if the Interlock Write is to an invalid location.

Register reads to unmapped register address spaces return all 0’s. Registers writes to unmapped register address spaces have no effect.

Simultaneous Read cycles on the Flash array and Read/Write cycles on the registers are possible, but not simultaneous register Read/Write accesses with Flash Array Interlock Writes are forbidden.

17.4.2.2 Error correction code

The flash memory module implements a strategy for improving the reliability of the data stored in Flash via the use of Error Correction Code.
In Code Memory the ECC size is 128 bits with 9 ECC bits associated. In Data Memory the ECC size is 64 bits with 15 ECC bits associated.
ECC function is implemented in such a way to guarantee Double Error Correction and Triple Error Detection (DEC-TED) in Data Memory.

ECC function is implemented in such a way to guarantee Single Error Correction and Double Error Detection (SEC-DED) in Code Memory.
ECC coding has all 0’s (intended as data and parity) as an invalid word and all 1’s as a valid word.

In addition to providing the requested Read data, the Flash also provides output sidebands reflecting the status of the single and double error corrections and triple error detections performed at the end of each Read.

For functional safety coverage, Flash performs an EDC post ECC operation at every Read to detect mismatches in the ECC logic detection and correction. The result is output as sideband. Also active is a monitor circuit that checks the reference current in the sense amplifiers, including control of the internally regulated Read voltage, to prevent anomalous Read operation.

ECC circuitry provides single-bit fault correction used to achieve automotive reliability targets. Some units experience single-bit corrections throughout the life of the product with no impact on product reliability.

17.4.2.2.1 Multi-bits error management (more errors than uncorrectable detection capability)

In this case, behavior cannot be predicted without knowing the original data and the position of the bit flips. The hardware can react in multiple ways:

1. do nothing (that is, not signaling of any flag)
2. assert either SBC, SBC1, or EER without EEE
3. assert either SBC, SBC1, or EER with EEE
4. assert EEE only.

Note: SBC and SBC1 reporting have to be activated in UT0.

It is worth highlighting that by design the purpose of the ECC logic is not to detect multi-bit errors but cope with errors in the data and parity containing maximum number of bit flip till uncorrectable detection capability. In this condition, the EDC after ECC monitors the integrity of the ECC logic.

Other mechanisms must detect multi-bit errors or avoid their occurrence.

For example running the Array Integrity Check once after the boot before the safety application starts to protect the Data with an application level checksum.

An example of a mechanism that decreases the possibility of multi-bit errors in the array is the multiplexing. The bits of the same word are not one next to each other, but they are spaced of a fixed number of bits.

17.4.2.2.2 EEPROM emulation

The chosen ECC allows some bit manipulation so that a doubleword can be re-written several times without needing an erase of the block. This allows the use of a doubleword to store flags useful for the EEPROM emulation.

The sequence provided in *Table 249: Bits Manipulation: doublewords preserving ECC integrity* is related to Data Memory (therefore would not apply on Code Memory). The ECC coding allows, starting from an all 1's doubleword value, to rewrite subsequent predefined code values changing only 1s into 0s and preserving the ECC integrity.

Caution: Predefined code values have to be applied following the exact Byte order, as reported in the table.

Table 249. Bits Manipulation: doublewords preserving ECC integrity

Doubleword	ECC parity bits	Sequence
B7-B6-B5-B4-B3-B2_B1-B0		
0xFFFF_FFFF_FFFF_FFFF	0x7FFF	erase
0xFFFF_FFFF_FFFF_FFFE	0x3C88	write1
0xFFFF_FFFF_FEFE_FC80	0x3C88	write2
0xFFFF_FFFC_FEFC_FC80	0x1880	write3
0xFCFC_FFFC_FCFC_F880	0x1880	write4
0xFCF8_F8F0_FCE0_E080	0x1800	write5
0xF8F8_E080_F0E0_E000	0x1000	write6
0xE0E0_E080_E0E0_C000	0x1000	write7
0xE000_8000_00C0_0000	0x1000	write8

17.4.2.3 Flash address generation check

For functional safety coverage, Flash provides output sidebands reflecting the re-encoded address used to perform the actual Read operation. In the re-encoding process both word line and column within the selected block are evaluated. For the completion of the re-encoding operation, the addresses related to block selections are also computed. Re-encoding circuitry is able to detect both no-selection as well as multi-selection. Flash provides a self-check of address consistency comparing at each Read the sampling of the current Read address with the re-encoded one, and in case of mismatch an error sideband is set.

17.4.3 Modify mode

All the Modify operations of the flash memory module are managed through the Flash User registers interface. This interface is referred as MAIN.

When a Modify operation is active on some blocks, no Read access is possible on any other block of the same Flash Partition. The Read-While-Modify is supported only among different Flash partitions.

During a Flash Modify operation any attempt to read any Flash location in the same Flash partition outputs invalid data and MCR.RWE is automatically set.

If a reset occurs during a Modify operation, the operation is immediately terminated and the module is reset to Read mode. The data integrity of the Flash section where the Modify operation was terminated is not guaranteed: the interrupted Flash Modify operation must be repeated.

In general each Modify operation is started through a sequence of 3 steps:

1. The first instruction is used to select the desired operation by setting its corresponding selection bit in MCR.PGM or MCR.ERS or UT0.MRE.
2. The second step is the definition of the operands: the address and the data for Programming, or the blocks for Array Integrity Check or Margin Read.
3. The third instruction is used to start the Modify operation by setting MCR.EHV or UT0.AIE.

Once selected, but not yet initiated, one operation can be canceled by resetting the operation selection bit.

A summary of the available Flash Modify operations is shown in *Table 250*.

Table 250. Flash Modify operations

Operation	Select bit	Operands	Start bit
Doubleword Program	MCR.PGM	Address and Data by Interlock Writes	MCR.EHV
Block Erase	MCR.ERS	SEL0, SEL1, SEL2 and SEL3	MCR.EHV
Array Integrity Check	None	SEL0, SEL1, SEL2 and SEL3	UT0.AIE
Margin Read	UT0.MRE	UT0.MRV + SEL0, SEL1, SEL2 and SEL3	UT0.AIE

Once MCR.EHV (or UT0.AIE) is set, all the operands can no longer be modified until MCR.DONE (or UT0.AID) is high.

In general each Modify operation is completed through a sequence of 4 steps:

1. Wait for operation completion: wait for MCR.DONE (or UT0.AID) to go high.
2. Check operation result: check bit MCR.PEG (or compare UM0–UM9 with expected value).
3. Switch-Off FPEC by resetting MCR.EHV (or UT0.AIE).
4. Deselect current operation by clearing MCR.PGM/ERS (or UT0.MRE).

Note: Do not initiate a Modify operation on a flash memory module while another Modify operation is in progress on a separate module.

In the following sections, all the possible Modify operations are described with accompanying examples of the sequences needed to activate them.

17.4.3.1 Program

A Flash Program sequence operates on any doubleword within the Flash core. Up to two words within the doubleword may be altered in a single doubleword program operation. A single program operation can also alter one page (4 doublewords, 32 bytes), or 4 pages max (128 bytes in total). It is possible to program only certain doublewords in the page or the quad-page, leaving the others with their original content. Whenever the array is programmed, the ECC bits are also programmed.

ECC is handled on a 64-bit boundary in Data Flash and 128b boundary in Code Flash. Whenever you program, ECC bits are also programmed. Thus, if only one word in any given ECC segment is programmed, the adjoining word/s (in that segment) should not be programmed later on, since ECC calculation has already completed for that segment. Attempts to program the adjoining word will probably result in an operation failure. All programming operations should be from 64 bits to 1024 bits, should be 64-bit aligned for Data Flash and 128-bit aligned for Code Flash. The programming operation should completely fill selected ECC segments.

Due to the fact that Flash functions in a System with an end2end ECC scheme, ECC parity bits are also input during an Interlock Write. After setting MCR.EHV, the FPEC checks the integrity of the data received in terms of ECC. If any doubleword in the doubleword program, the page program or the quad-page program contains an inconsistency between data and parity, the whole operation is rejected and a MCR.PEG=0 is set. The doublewords without data and with parity inconsistency are thus not programmed.

Programming changes the value stored in an array bit from logic ‘1’ to logic ‘0’ only. Programming cannot change a stored logic ‘0’ to a logic ‘1’. If a logic ‘0’ is attempted to be over programmed by a logic ‘1’, the resulting operation fails (MCR.PEG=0), and the 0s that are interlocked are merged (ORed) with 0s that are already present in the 64-bit ECC segment. Extreme care has to be taken when doing such an operation due to the high risk of producing ECC errors.

Addresses in locked blocks cannot be programmed. Register locked blocks cannot be programmed. Hardware locking is also available which only affects the program operation. Hardware locking is enabled by sideband signals, and if enabled, programs are prevented from accessing hardware locked blocks in the same way as program software locking.

The user may program the values in any or all the eight words in a page with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. Only certain doublewords may be programmed within the page or the quad-page, without altering the content of the ones unselected via interlock.

The Program operation consists of the following sequence of events:

1. Change the value of MCR.PGM from ‘0’ to ‘1’.
2. Ensure the block that contains the address to be programmed is unlocked. Write the first address to be programmed with the program data. The flash memory module latches address bits (23:7) at this time, as well as written data. This Write is referred to as a Program Data Interlock Write. An Interlock Write may be as large as 64 bits, and as small as 32 bits (depending on the CPU bus).
3. If more than 1 word is to be programmed, write the additional address in the doubleword with data to be programmed. This is referred to as a Program Data Write. The flash memory modules ignore address bits (23:7) for program data writes. The eventual unwritten data word defaults to 0xFFFFFFFF.
4. Write a logic ‘1’ to the MCR.EHV to start the internal program sequence or terminate via step 9.
5. Wait until the MCR.DONE goes high.
6. Confirm MCR.PEG=1.
7. Write a logic ‘0’ to the MCR.EHV.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic ‘0’ to the MCR.PGM to terminate the program operation.

Programs may be initiated with the 0–1 transition of MCR.PGM or by clearing MCR.EHV at the end of a previous program. The first Write after a program is initiated determines the page address to be programmed. This first Write is referred to as an Interlock Write. The Interlock Write determines if the UTEST or normal array space will be programmed via MCR.PEAS.

In the case of an erase-suspended program, the values in MCR.PEAS may be modified via the Program Interlock Write, enabling erase-suspended programs to UTEST Nonvolatile Memory space and revert back to their original state (related to erase-suspended operation) once the erase-suspended program is completed.

An Interlock Write must be performed before setting MCR.EHV. An Interlock Write performed after setting MCR.EHV is forbidden and data and address passed through interlock is lost.The user may terminate a program sequence by clearing MCR.PGM prior to setting MCR.EHV.

After the Interlock Write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024 bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of 0xFFFF_FFFF. If multiple writes are performed on the same location, the data for the last Write is used in programming.

While MCR.DONE is low and MCR.EHV is high, the user may clear EHV, resulting in a Program Abort.

A Program Abort forces the module to step 8 of the program sequence.

An aborted program results in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The data space being operated on before the Abort contains indeterminate data. This may be recovered by repeating the same program instruction or executing an Erase of the affected blocks.

The user may not abort a program sequence while in Program Suspend.

Figure 174. Doubleword program of data 0x55AA_55AA at address 0x00_AAA8 and data 0xAA55_AA55 at address 0x00_AAAC

```
MCR = 0x0000_0010;           /* Set PGM in MCR: Select Operation */
(0x00_AAA8) = 0x55AA_55AA;   /* Latch Address and 32 LSB data */
(0x00_AAAC) = 0xAA55_AA55;   /* Latch 32 MSB data */
MCR = 0x0000_0011;           /* Set EHV in MCR: Operation Start */
do                           /* Loop to wait for DONE=1 */
{ tmp = MCR;                 /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
status = MCR & 0x0000_0200;  /* Check PEG flag */
MCR = 0x0000_0010;           /* Reset EHV in MCR: Operation End */
MCR = 0x0000_0000;           /* Reset PGM: Deselect Operation */
```

17.4.3.1.1 Program Suspend/Resume

The program sequence may be suspended to allow Read access to the Flash array. It is not possible to program or to erase during a Program Suspend. Read-While-Write operations may also be used to read the array during a program sequence providing the Read is to a different partition.

During Program Suspend, all reads to the doubleword targeted for program return indeterminate data.

A Program Suspend can be initiated by changing the value of MCR.PSUS from ‘0’ to ‘1’. MCR.PSUS can be set to ‘1’ at any time when MCR.PGM and MCR.EHV are high and MCR.ERS is low. 0–1 transition of MCR.PSUS causes the module to start the sequence which places it in Program Suspend.

It is possible to suspend a program performed during an Erase Suspend. The user must wait until MCR.DONE=1 before the module is suspended and further actions are attempted. MCR.DONE takes no more time than t_{PSUS} to go high after MCR.PSUS is set to ‘1’.

Once suspended, the array may be read. Flash memory reads while MCR.PSUS=1 from the doubleword being programmed return indeterminate data.

Figure 175. Doubleword Program Suspend

```
MCR = 0x0000_0019; /* Set PSUS in MCR: Program Suspend */
do                 /* Loop to wait for DONE=1 */
{ tmp = MCR;       /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
```

Note that there is no need to clear MCR.EHV and MCR.PGM in order to perform reads during program suspend.

The Program sequence is resumed by writing a logic ‘0’ to MCR.PSUS.

MCR.EHV must be set to ‘1’ before MCR.PSUS can be cleared to resume the operation.

The module continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

Figure 176. Doubleword Program Resume

```
MCR = 0x0000_0011;/* Reset PSUS in MCR: Program Resume */
```

Repeated suspends at a high frequency may result in the operation timing out, and the embedded flash memory responds by completing the operation with a fail code (MCR.PEG=0). The minimum time between Program Suspends to ensure this does not occur is t_{PSRT}.

17.4.3.1.2 Over-programming protection (OPP) enable

Over Programming Protection can be enabled on a block-by-block basis. In an OPP enabled block of code Flash, any quad-word which has already been programmed cannot be programmed again. The OPP enable does not affect the Erase operation. It is possible to independently set OPP and (hardware) erase protection in order to avoid any kind of overwrite. Using these protection mechanisms in conjunction with the Test mode disable feature gives even greater overwrite protection. Attempts to over-program result in MCR.PEG being cleared.

OPP can be enabled for 16 KB, 32 KB, 64 KB and 256 KB blocks.

OPP is enabled by sideband signals and, if enabled, the program operation is modified to check for programmed bits prior to undertaking a high voltage program event.

17.4.3.1.3 Successful program address

To enable the System to create logic to authenticate operations with a program (like in a diary operation), the flash memory module provides the last successful address programmed as a sideband signal. This value is held until the next successful program. A successful program is defined as a program operation of data that is not all 1s, to an unlocked block and the operation results in MCR.PEG returning a ‘1’. An all 1’s value is the default (and invalid) address after reset.

In case of page programming, the last successful address returned is the beginning of the page itself regardless of whether that specific address has not been passed through interlock. In case of quad-page programming, the last successful address returned is the beginning of the quad-page itself regardless of whether that specific address has not been passed through interlock.

17.4.3.2 Erase

Erase changes the value stored in all bits of the selected block(s) to logic ‘1’. An Erase sequence operates on any combination of blocks in the low, mid, high or 256K address space. The Test block cannot be erased.

The Erase sequence is fully automated within the Flash. The user only needs to select the blocks to be erased and initiate the Erase sequence.

Register locked blocks cannot be erased. Hardware locking, which only affects the Erase operation is also available. Hardware locking is enabled by sideband signals and, if enabled, erases are prohibited on hardware locked blocks in the same way as erase software locking. If multiple blocks are selected for erase during an Erase sequence, no specific operation order can be assumed.

The Erase operation consists of the following sequence of events:

1. Change the value in the MCR.ERS bit from ‘0’ to ‘1’.
2. Select the block(s) to be erased by writing 1’s to the appropriate register(s) in SEL0, SEL1, SEL2 and SEL3 registers. Note that Lock and Select are independent. If a block is selected and locked, no erase occurs.
3. Write to any address in Flash. This is referred to as an Erase Interlock Write.
4. Write a logic ‘1’ to MCR.EHV to start the internal Erase sequence or skip to step 9 to terminate.
5. Wait until MCR.DONE goes high.
6. Confirm MCR.PEG=1.
7. Write a logic ‘0’ to MCR.EHV.
8. If more blocks are to be erased, return to step 2.
9. Write a logic ‘0’ to the MCR.ERS bit to terminate the Erase operation.

After setting MCR.ERS, one Write (referred to as an Interlock Write) must be performed before MCR.EHV can be set to ‘1’.

Data words written during Erase sequence Interlock Writes are ignored. As Flash functions in a System with an end2end ECC scheme, ECC parity bits are also input during an Interlock Write. After setting MCR.EHV, the FPEC checks the integrity of the data received in terms of ECC even if the value of the data is not meaningful for the Erase operation. If an inconsistency between data and parity is found, the whole operation is rejected and MCR.PEG=0 is set.

The user may terminate the Erase sequence by clearing ERS before setting EHV.

An Erase operation may be aborted by clearing MCR.EHV assuming MCR.DONE is low, MCR.EHV is high and MCR.ESUS is low. An Erase Abort forces the module to step 8 of the Erase sequence. An aborted Erase results in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The block(s) being operated on before the Abort contain indeterminate data. This may be recovered by executing an Erase on the affected blocks. The user may not abort an Erase sequence while in Erase-Suspend.

Figure 177. Erase of blocks (example)

```
MCR = 0x0000_0004;            /* Set ERS in MCR: Select Operation */
SEL0 = 0x0006_0000;           /* Set the corresponding bit in SEL0: Select
Blocks */
(0x00_0000) = 0xFFFF_FFFF;    /* Latch a Flash Address */
MCR = 0x0000_0005;            /* Set EHV in MCR: Operation Start */
do                            /* Loop to wait for DONE=1 */
{ tmp = MCR;                  /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
status = MCR & 0x0000_0200;   /* Check PEG flag */
MCR = 0x0000_0004;            /* Reset EHV in MCR: Operation End */
```

17.4.3.2.1 Erase Suspend/Resume

The Erase sequence may be suspended to allow Read access to the Flash array. It is possible to program another block in any flash memory module during Erase-Suspend.

Programming a location subject to suspended Erase results in the program not being executed and PEG being set to 0.

It is not possible to erase any block during an Erase-Suspend.

During Erase-Suspend, all reads to blocks targeted for Erase return indeterminate data.

An Erase-Suspend can be initiated by changing the value of MCR.ESUS from ‘0’ to ‘1’. MCR.ESUS can be set to 1 at any time when MCR.ERS and MCR.EHV are high and MCR.PGM is low. A 0–1 transition of MCR.ESUS causes the module to start the sequence which places it in Erase-Suspend. The User must wait until MCR.DONE=1 before the module is suspended and further actions are attempted.

MCR.DONE takes no longer than t_{ESUS} to go high after MCR.ESUS is set to ‘1’.

Once suspended, it is not possible to alter the content of SEL0, SEL1, SEL2 and SEL3 registers. Once suspended, the array may be read. Flash memory reads while MCR.ESUS=1 from the block(s) being erased return indeterminate data.

Figure 178. Block Erase Suspend

```
MCR = 0x0000_0007; /* Set ESUS in MCR: Erase Suspend */
do                 /* Loop to wait for DONE=1 */
{ tmp = MCR;       /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
```

Note that there is no need to clear MCR.EHV and MCR.ERS in order to perform reads during Erase-Suspend.

The Erase sequence is resumed by writing a logic ‘0’ to MCR.ESUS.

MCR.EHV must be set to ‘1’ before MCR.ESUS can be cleared to resume the operation. The module continues the Erase sequence from one of a set of predefined points and this may extend the time required for the Erase operation.

Figure 179. Block Erase Resume

```
MCR= 0x0000_0005;/* Reset ESUS in MCR: Erase Resume */
```

Repeated suspends at a high frequency may result in the operation timing out and the embedded flash memory responds by completing the operation with a fail code (MCR.PEG=0). The minimum time between Erase suspends to ensure this does not occur is t_{ESRT}.

17.4.3.3 Factory Mode

The Factory Mode feature can be used to improve Program and Erase Times and it is managed through a diary location mechanism. If location UTEST at offset +0x0020 is found virgin at the moment of the reset, MCR.FERS can be set, after rising MCR.ERS and before MCR.EHV. After setting MCR.EHV the FPEC recognizes the value of FERS and allows a fast Program and Erase Operation. Under those conditions a factory Mode can start and no Read-While-Write is allowed (all the flh_busy sideband signals are set high). If the diary location has been programmed by the user (having changed from being virgin at reset), the Factory Mode gets permanently disabled.

17.4.4 Alternate program and erase interface

A second Modify interface is provided on the Flash for program and erase operations. The intent is that a dedicated master can have private access to blocks through this interface.

The alternate interface includes an alternate MCR register and alternate lock registers. Any block in low, mid, or high address space may be accessible through the alternate interface. The UTest NVM Block is also accessible for program through the alternate interface. Blocks in 256K Address Space are not accessible through this interface.

The alternate interface supports programming of data sizes from one doubleword (64 bits) to four doublewords within a single page (256 bits). Within a page, up to eight words may be altered in a single program operation. It is recommended that programming operations through the alternate interface be from 64 bits to 256 bits, and be 64-bit aligned. Programming has the same constraints as for the Main interface concerning ECC boundaries: 64b for Data Flash and 128b for Code Flash.

The alternate interface supports erasing a single block at a time with each erase operation.

As with the main interface, erase suspend, program suspend, and erase-suspended programs are supported.

Program and Erase times for the alternate interface are the same as the main interface when programs and erases are done in isolation. If the main interface and alternate interface are being utilized simultaneously, bandwidth sharing is utilized.

Note: It is possible for the UTest NVM block to exist in both the main program interface and alternate program interface. In the event of a conflict the first interface to do the program to an address will get priority, and all other attempts after this will result in an OTP over-program failure (MCR[PEG] or MCRA[PEGa] being set to 0).

17.4.4.1 Alternate program interface

The programming procedure follows exactly the same flow as the main interface, except that only page programming is allowed.

The alternate interface program is done at lower priority compared to the main interface. Based on the size of programming allowed, the main interface has a 4:1 ratio advantage over the alternate interface. Program operations are allowed to finish on the active interface prior to allowing the inactive interface to execute its operation. Suspends done on the active interface enable the inactive interface to begin its operation.

For alternate program while erase, there could be a latency of up to 5 ms in the program operation.

For more information on interaction between the alternate program interface and the main interface, see *Section 17.4.4.3: Alternate interface performance*.

If the alternate interface is the only interface active, then normal program times may be expected.

17.4.4.2 Alternate erase interface

For erases done through the alternate interface, the single block to be erased is selected based on the address that is received during the erase interlock write. Thus the Select registers are not valid for the alternate interface (step 2 of the erase flow in *Section 17.4.3.2: Erase*), and the interlock write address is used (step 3 of the erase flow in *Section 17.4.3.2: Erase*). All other features of the main interface apply to the alternate erase interface. The

erase procedure is exactly the same as for the main interface. See *Section 17.4.3.2: Erase* for more information on erase.

The alternate interface erase is done at lower priority compared to the main interface. The time-slice ratio between the two interfaces is 10:1, with 50 ms of time given to the main erase for every 5 ms of alternate interface. Suspends done on the active interface enable the inactive interface to begin its operation.

For alternate erase while program, there can be a latency of up to 5 ms in the main program time. Once the alternate erase is interrupted, it will remain interrupted for main programs for a fixed time of 50 ms, at which time the alternate erase will restart. It should be noted that multiple interruptions of the alternate erase will significantly increase the total erase time.

For alternate erase while erase, the main erase has priority and receives 50 ms of erase for every 5 ms of alternate erase. This may result in a significant increase in the erase time for the alternate interface, and a minor increase in erase time for the main interface.

For more information on interaction between the alternate erase interface and the main interface, see *Section 17.4.4.3: Alternate interface performance*.

If the alternate interface is the only interface active, then normal erase times may be expected.

17.4.4.3 Alternate interface performance

Table 251 highlights the interactions between the alternate interface and the main interface for program, erase, suspend, and idle conditions.

Table 251. Alternate Program and Erase Characteristics

Alt PGM/ERS interface request	Main PGM/ERS interface request	Alt interface result on memory array	Main interface result on memory array
Idle	Idle	Idle	Idle
Idle/SUSP	PGM/ERS	Idle/SUSP	PGM/ERS
PGM/ERS	Idle/SUSP	PGM/ERS	Idle/SUSP
PGM	PGM	PGM	Priority PGM (4:1 ratio)
PGM	ERS	PGM with 5ms latency	ERS (possible minimal increase to total ERS time)
ERS	PGM	ERS (significant increase in total ERS time expected)	Priority PGM (possible latency of 5 ms) Once PGM is granted, 50 ms time slice available for more programs
ERS	ERS	ERS (significant increase in total ERS time expected)	Priority ERS (10:1 ratio) (possible minimal increase to total ERS time) 50 ms Main ERS for every 5 ms alternate ERS

17.4.5 User Test mode

Customers can put the flash memory module in User Test mode to perform specific tests for the integrity of the Flash array.

Three kinds of test can be performed:

- Array Integrity Self Check
- Margin Read
- Vth Distribution

User Test mode is an exclusive operation—it cannot be run in conjunction with any other Flash mode (Read and Write), and Read-While-Write functionality is inapplicable. Read accesses attempted by the user during User Test mode generates a Read-While-Write Error (RWE of MCR set). Vth Distribution mode is conditioned by SoC Life Status (mode allowed only as Failure analysis)

It is not permitted to perform User Test operations on the Test blocks (including the UTEST section).

17.4.5.1 Array Integrity Self Check

Array integrity is checked using a predefined address sequence (proprietary), and this operation is executed on selected (SEL0, SEL1, SEL2 and SEL3) blocks. Blocks marked by TMDBS are automatically excluded from MISR signature computation. Any random or non-random code is valid. Once the operation is completed, the results of the reads can be checked by reading the MISR value (stored in UM0–UM9), to determine if an incorrect Read, or ECC detection was noted. Array integrity requires that the Read Wait States and Address Pipelined control registers in the BIU be set to match the frequency being used.

The internal MISR calculator is a 64 + 8 + 1-bit register.

The 256 data, the respective ECC parity data, and the uncorrectable ECC (ERR) errors of the four doublewords are therefore captured by the MISR through 4 different Read accesses at the same location.

The whole check is done through 4 complete scans of the memory address space:

1. The first pass scans only bits 63-0 + 8 ecc + ERR of each doubleword.
2. The second pass scans only bits 128-64 + 8 ecc + ERR of each doubleword.
3. The third pass scans only bits 191-129 + 8 ecc + ERR of each doubleword.
4. The fourth pass scans only bits 255-192 + 8 ecc + ERR of each doubleword.

The 256-bit data and the respective ECC parity data are sampled after any single-bit/double-bit ECC correction and/or the uncorrectable error flag is set after any ECC detection. Single-bit corrections are always active and logging may be enabled (by UT0.SBCE1), resulting in the MCR.SBC1 flags and related address to be captured in ADR register, but this does not change the final signature in the UM0–UM9.

Only data from existing locations are captured by the MISR. The MISR can be seeded to any value by writing the UM0–UM9 registers.

The Array Integrity Self Check consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1’s to the appropriate field(s) in SEL0, SEL1, SEL2 and SEL3 registers. Blocks selected for Array Integrity Check do not need

to be unlocked. It is not possible to perform Array Integrity operations on the UTEST Nonvolatile Memory block.

3. UT0.AIS must be set to one.
4. Seed the MISR UM0–UM9 with desired values.
5. If breakpoints are desired, set UT0.AIBPE to ‘1’, and ensure that MCR.EER and MCR.SBC1 are cleared. If breaking on single-bit correction is required, ensure that UT0.SBCE1 is set.
6. Write a logic ‘1’ to UT0.AIE to start the Array Integrity Check. Array integrity operations may be aborted prior to UT0.AID going high. This may be done by clearing UT0.AIE. It should be noted that in the event of an aborted Array Integrity check, the MISR registers contain a signature for the portion of the operation that was completed prior to the Abort, and is not definitive. Prior to performing further Array Integrity operations, the UM0–UM9 registers may require initialization to the desired seed value by performing register writes. Array Integrity operations may be suspended while UT0.AID is low. UT0.AISUS may be set to request an Array Integrity Suspend. Once UT0.AID has been set high, UT0.AISUS may be cleared to resume the Array Integrity sequence.
7. Wait until UT0.AID goes high.
8. Check that UT0.NAIBP is set to ‘1’ if breakpoints were previously enabled so that the MCR.EER, MCR.SBC1 and ADR registers may be checked to determine the cause and address of the break. Prior to resuming the operation, it is necessary to clear MCR.SBC1 or MCR.EER in order to preserve coherency of the flags and ADR updates with respect to any subsequent breakpoints. Operation may then be resumed by clearing UT0.NAIBP. Continue waiting until the UT0.AID bit goes high. If breakpoints were not enabled, or if UT0.NAIBP is low when UT0.AID goes high, then the operation is complete.
9. Compare UM0–UM9 content with the expected result.
10. Write a logic ‘0’ to the UT0.AIE bit.

Caution: Do not modify the content of Block Select (SEL0, SEL1, SEL2 and SEL3) and Lock (LOCK0, LOCK1, LOCK2 and LOCK3) registers during the execution of the Array Integrity Check operation or the MISR value may vary unpredictably.

User mode array reads requested during the Array Integrity test are ignored to ensure that the Array Integrity operation is not corrupted. The memory array does not respond to array Read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

While UT0.AID is low and UT0.AIE is high, the user may clear AIE, resulting in an Array Integrity Check Abort. UT0.AID must be checked to know when the aborting command has completed.

Array Integrity Check can be suspended by setting UT0.AISUS while Array Integrity Check is running (UT0.AID=0). Once suspended (UT0.AID=1) Array Integrity Check cannot be aborted and clearing UT0.AISUS starts the Resume.

Figure 180. Array Integrity Check of blocks (example)

```
UM0/9 = 0x0000_0000; /* Reset UM0/9 content */
UT0 = 0xF9F9_9999;   /* Set UTE in UT0: Enable User Test */
SEL0 = 0x0006_0000;  /* Set the corresponding bit in SEL0: Select Blocks
*/
SEL2 = 0x0000_0006;  /* Set the corresponding bit in SEL2: Select Blocks
*/
UT0 = 0x8000_0002;   /* Set AIE in UT0: Operation Start */
do                   /* Loop to wait for AID=1 */
{ tmp = UT0;         /* Read UT0 */
} while ( !(tmp & 0x0000_0001) );
```

17.4.5.2 Margin Read

User Margin Read may be achieved using the Array Integrity interface and has all the associated features of the Array Integrity interface (MISR calculation, suspend capability and Breakpoints).

Margin Read procedure (either Margin 0 or Margin 1) can be run on selected blocks in order to unbalance the Sense Amplifiers with respect to standard Read conditions, so that all the Read accesses reduce the margin vs ‘0’ (UT0.MRV=0) or vs ‘1’ (UT0.MRV=1).

User Margin Read is a self timed event and is independent of system clocks or wait states selected. Margin Read can be run on selected (SEL0, SEL1, SEL2 and SEL3) blocks, generating checksum calculations on MISR and ECC flagging. Blocks marked by TMDBS are automatically excluded from MISR signature computation. Once the operations are completed, the results of the Margin Reads can be checked, comparing checksum values in UM0–UM9 and reading MCR.EER and MCR.SBC to determine if zero, one or two bits are being detected by the Margin Read.

The MISR can be seeded to any value by writing the UM0–UM9 registers.

As with Array Integrity Check, the internal MISR calculator is a 64 + 8 + 1-bit register. The 256-bit data, the corresponding ECC parity data and the uncorrectable ECC (ERR) errors of the four doublewords are therefore captured by the MISR through 4 different Read accesses at the same location. The whole check is accomplished through 4 complete scans of the memory address space:

1. The first pass scans only bits 63-0 + 8 ecc + ERR of each doubleword.
2. The second pass scans only bits 128-64 + 8 ecc + ERR of each doubleword.
3. The third pass scans only bits 191-129 + 8 ecc + ERR of each doubleword.
4. The fourth pass scans only bits 255-192 + 8 ecc + ERR of each doubleword.

The 256-bit data and the respective ECC parity data are sampled after any single-bit ECC correction, and the double error flags are set after any ECC detection. Single-bit corrections are always active and logging may be enabled (by UT0.SBCE1), resulting in the MCR.SBC1 flags and related address to be captured in ADR register, but this does not change the final signature in the UM0–UM9.

In any case, the charge losses detected through the Margin Read cannot be considered device failures and no Failure Analysis is initiated on them.

The Margin Read Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in SEL0, SEL1, SEL2 and SEL3 registers. Blocks selected for Margin Read do not need to be unlocked. It is not possible to do Margin Read operations on the UTEST Nonvolatile Memory block.
3. Seed the MISR UM0–UM9 with desired values.
4. Ensure that MCR.EER and MCR.SBC1 are cleared.
5. Set UT0.SBCE1 to '1' to detect single bits during Margin Read.
6. Change the value of UT0.MRE from '0' to '1'.
7. Select the Margin level: UT0.MRV=0 for 0's margin, UT0.MRV=1 for 1's margin.
8. Write a logic '1' to UT0.AIE to start the Margin Read.
9. Wait until UT0.AID goes high.
10. Check that UT0.NAIBP is set to '1' if breakpoints were previously enabled so that MCR.EER, MCR.SBC and ADR.ADDR may be checked to determine the cause and address of the break. Prior to resuming operation, clear MCR.SBC1 or MCR.EER. Operation may then be resumed by clearing UT0.NAIBP. Continue to wait until UT0.AID goes high. If breakpoints were not enabled, or if UT0.NAIBP is low when UT0.AID goes high, then the operation is complete.
11. Compare UM0–UM9 content with the expected results.
12. Check MCR.EER and MCR.SBC1.
13. Write a logic '0' to UT0.AIE, UT0.MRE and UT0.MRV.
14. If more blocks are to be checked, return to step 2.

The linear address sequence is used regardless of the value of UT0.AIS.

Caution: Do not modify the content of Block Select (SEL0, SEL1, SEL2 and SEL3) and Lock (LOCK0, LOCK1, LOCK2 and LOCK3) registers during execution of the Margin Read operation, otherwise the MISR value may vary unpredictably.

User mode array reads requested during the Margin Read are ignored to ensure that the Margin Read operation is not corrupted. The memory array does not respond to array Read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

While UT0.AID is low and UT0.AIE is high, the user may abort Margin Read by clearing AIE. UT0.AID must be checked to know when the aborting command has completed. Margin Read can be suspended by setting UT0.AISUS while Margin Read is running (UT0.AID=0). Once suspended (UT0.AID=1), Margin Read cannot be aborted and clearing UT0.AISUS starts the Resume.

Figure 181. Margin Read Check versus 1's

```
UM0/9 = 0x0000_0000; /* Reset UM0/9 content */
UT0 = 0xF9F9_9999;   /* Set UTE in UT0: Enable User Test */
SEL0 = 0x0006_0000;  /* Set LSL2-1 in SEL0: Select Blocks */
UT0 = 0x8000_0020;   /* Set MRE in UT0: Select Operation */
UT0 = 0x8000_0030;   /* Set MRV: Select Margin versus 1's */
UT0 = 0x8000_0032;   /* Set AIE in UT0: Operation Start */
do                   /* Loop to wait for AID=1 */
{ tmp = UT0;         /* Read UT0 */
} while ( !(tmp & 0x0000_0001) );
data0/9 = UM0/9;     /* Read UM0/9 content*/
UT0 = 0x8000_0030;   /* Reset AIE in UT0: Operation End */
UT0 = 0x0000_0000;   /* Reset UTE, MRE, MRV, AIS in UT0:
                        Deselect Operation */
```

17.4.5.3 Vth Distribution

Vth Distribution functionality can be used in case of Failure Analysis for quick assessment of cells status in terms of Vth. The functionality has bit register UTO.VTD as initiator. Writing of this bit is enabled by the Life Cycle state of the device. Life Cycles has to be in Failure Analysis (FA) condition to allow Vth Distribution mode.

Vth Distribution functionality works on selected blocks on the base of multi internal runs. Each run corresponds to an internal voltage step, where cells read as '1' (Vth lower than internal voltage level) are counted: the completion of each run provides on UMISR0 the count of these cells. Clearing UTO.NAIBP starts next voltage step run, till completion of the intended voltage range. Number of internal voltage step is fixed. Aim of the test is to sweep the range from the erased cells with the highest Vth to the programmed cells with the lowest Vth, therefore to provide an assessment of the effective distribution of the cells in the selected blocks.
Once all the cell counts, one per run, have been collected, distribution can be obtained as difference between counts collected through subsequent steps.
In every run, all the cells, including the ECC parity ones, are counted for the block selected.
Vth Distribution results are expected to be collected at room temperature (25C).

Vth Distribution Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in SEL0, SEL1, SEL2 and SEL3 registers.
3. Unlock blocks to be checked by writing 0's to the appropriate register(s) in LOCK0, LOCK1, LOCK2 and LOCK3 registers. Blocks selected for Vth Distribution need to be

unlocked. It is not possible to do Vth Distribution operations on the UTEST Nonvolatile Memory block, neither on blocks protected through TMD SEAL Block Select.

4. Set UT0.VTD to ‘1’ to select Vth Distribution mode.
5. Write a logic ‘1’ to UT0.AIE to start the first internal run.
6. Wait until UT0.AID goes high.
7. Check that UT0.NAIBP is set to ‘1’. If NAIBP is at ‘1’ it means current internal run has been completed and device is ready to increase voltage step. If NAIBP is at ‘0’ the overall sweep has been completed.
8. Collect UM0 content for the current voltage step (run).
9. Clear UT0.NAIBP to increase voltage step and return to 6)

Caution: Voltage levels used during Vth distribution sweep may alter the intended reliability of the Cells as data retention. For this reason Vth distribution can be used no more than 10 times.

17.4.6 Protection strategy

Two kinds of protection are available: Modify Protection to avoid unwanted program/erase in Flash blocks and Test Mode Disable to avoid piracy.

17.4.6.1 Modify protection

The Flash Modify Protection information is stored in nonvolatile Flash cells located in the UTEST region. This information is read once during the Flash Initialization phase following exit from reset and is stored in volatile registers that act as actuators at SoC level. The information is passed to the flash memory module through erase_lock and program_lock sidebands, and allows independent protection of each block of Low, Mid, High and 256K Address Space Block.

There are also Address Space Block Locking registers provided to independently lock/unlock each block of Low, Mid, High and 256K Address Space Block against program and erase. Locking is done through the LOCK0, LOCK1, LOCK2 and LOCK3 registers. These registers do not have a nonvolatile image stored in the UTEST area of TestFlash, so the locking information is kept on reset: volatile content is all 1’s, meaning all blocks are locked.

All the volatile registers can be written to ‘0’ or ‘1’ at any time, therefore the user application can lock and unlock blocks when desired. Note, however, that overall protection is achieved by the combination of sidebands with Low, Mid, High and 256K Address Space Block Locking registers in such a way that a block can be programmed/erased only if both (sidebands and Locking registers) indicate an unlocked state. Sidebands independently control program protection, erase protection and over-programming protection (OPP) on a block-by-block basis.

As UTEST and BAF (each one with its own lock bit, LOCK0.TSLOCK and LOCK0.LOWLOCK respectively) are two sections of the same physical block (TestFlash), effective locking is achieved via the ‘AND’ operand of the lock bits. For effective locking, overall protection is obtained in conjunction with sidebands: UTEST and BAF may have dedicated program protection sidebands (System specific).

17.4.6.2 Test Mode Disable

In the TestFlash area a mechanism is also available to disable entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected for

protection in this way cannot be analyzed for possible failures by manufacture failure analysts.

Protection of this sort prevents all high voltage operations to the Flash executed by the internal FPEC as well as reads from FPEC, including Array Integrity Check or Margin Read, when using Test mode interfaces. Protection through other interfaces is managed through normal User mode protection mechanisms, that are System specific.

To disable manufacturer entry into Test mode, first program the Test Mode Disable Seal location to 0x2D3C_4B5A and once the next reset is asserted, Test mode is disabled.

It is possible to create a password to enable manufacturer entry into Test mode. This can be programmed into the Test Mode Disable Override Passcode. It is safest to do this prior to censoring the part to avoid unintended lockout. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and are not accepted for override. If the customer desires that override never be possible, one of the three invalid passcodes should be put into this location. Passcodes may be entered to authenticate entry (if enabled) by performing a register Write to offset address 0x90.

Only blocks selected in the Test Mode Disable Block Select field are controlled by the Test Mode Disable feature. Thus it is possible for customers to select blocks for this type of protection, and render them ineligible for manufacturer failure analysis. Bits programmed to ‘0’ in the Test Mode Disable Block Select field designate blocks that are controlled by Test Mode Disable feature. The TestFlash block, and thus the UTEST and BAF sections, is always protected once the Test Mode Disable Seal password is written. In order to permit two opportunities to select blocks for Test Mode disable, two regions are available, both in UTEST: one at offset 0x50, and the other at offset 0x60; both are 128-bit sized (only some may be effective, depending on actual sectorization). The bits of these two regions are logically ANDed in order to define which blocks are effectively selected for the Test Mode Disable feature.

The bit Block Select field is organized as follows, and aligned in the way blocks are defined in the LOCKx/SELx registers:

Table 252. Test Mode Disable block select

Block	Block selection bits
Blocks in Mid space	Data[15:0]
Blocks in Low space	Data[29:16]
Blocks in High space	Data[47:32]
Blocks in 256K space	Data[127:64]

18 Boot assist flash (BAF)

18.1 Introduction

The BAF code is programmed by ST. The two main tasks of BAF are to provide a serial bootloader feature and to search for the application entry point.

For details regarding how the BAF code execution is triggered, refer to Clock reset and boot chapter.

The BAF is executed via the IRCOSC as clock source.

The MCU is booted through a collaboration of several blocks, hardware and firmware. The first boot phases are performed by a state machine inside the System Status and Configuration Module (SSCM). Once completed, the SSCM sends a reset vector to the HW boot core of the device pointing into the Boot Assist Flash (BAF).

The BAF reads on special RCC registers to know: if a boot address has been updated, if the core2 is in lockstep and if the core2 boot address has been updated, it enables the core2 if needed, and it boots the application code in internal Flash.

If no Core1 User VTOR Address DCF record is found in internal flash memory and lifecycle is < “IN_FIELD”, BAF downloads application code serially using the UART module or the M_CAN module.

18.2 Memory map

For the BAF location in flash, refer to the Embedded memories chapter. The BAF block base address is 0x1FF0_0000. It is one time programmable (OTP) and is programmed during factory test. The address space and memory used by the BAF code are shown in *Table 253*.

Note: The BAF Flash block is assigned as OTP and it is protected from programming by a bit in the PASS LOCK0 registers; but the BAF Flash block can further be protected from programming by an external hardware pin. If the Hardware Interlock feature is enabled, while this pin is logic ‘0’ the BAF Flash block cannot be programmed.

Table 253. BAF memory organization

BAF component	Address
BAF image header	0x1FF0_0000
BAF code entry point	0x1FF0_0100
BAF data area and stack	0x5C00_0000–0x5C00_0603

18.2.1 BAF image header

BAF image contains a 256-byte header starting from address 0x1FF0_0000. The BAF header is explained in *Table 254*.

Table 254. BAF image header

Address offset	Size (bits)	Field description
0x0000	32	BAF version number
0x0004–0x000F	Reserved	
0x0010	32	Clock Jitter Activation Constant
0x0014–0x00FF	Reserved	

18.2.1.1 BAF image version

The BAF version is a 32-bit field in the image header starting at address 0x1FF0_0000. The 32-bit field is explained in *Table 255*.

Table 255. BAF image version

Field	Description
31:24	Major Number: This field contains the major version number for the BAF image
23:16	Minor Number: This field contains the minor version number for the BAF image
15:0	Reserved

18.2.1.2 Clock jitter activation constant

The Clock Jitter Activation constant is a 32-bit flag, but the smallest element in flash that can be programmed is 64 bits, so the first 32 bits are reserved to allow programming of this constant. This 32-bit flag can be programmed by the customer and is used to activate the Clock Jitter feature that adds clock edge jitter to the baud rate clocks of several communications modules.As part of the BAF startup procedure, BAF copies this constant from flash to the PASS Clock Jitter Enable Register.

Table 256. BAF Clock Jitter Activation Constant

Field	Description
0:31	Clock Jitter Activation Constant. The initial state of production disable flag is erased - 0xFFFF_FFFF. This equates to NO clock jitter.

18.3 Functional description

A functional description of the BAF modes is contained in the following sections.

18.3.1 Boot modes

Depending on the state and the content of its flash memory, the device can enter one of these different boot modes.

- Application code execution

  If the internal flash contains application code with a valid Core1 User VTOR Address DCF record, the application is executed.

- Serial boot

  If no valid Core1 User VTOR Address DCF record is found and the current life cycle phase of the device is less than Infield, an attempt is made to download the application by a serial protocol using the UART module or M_CAN module. The downloaded code is then executed by the boot CPU.

- Static mode

  The device enters power safe mode and waits for an external reset.

18.3.2 Device initialization

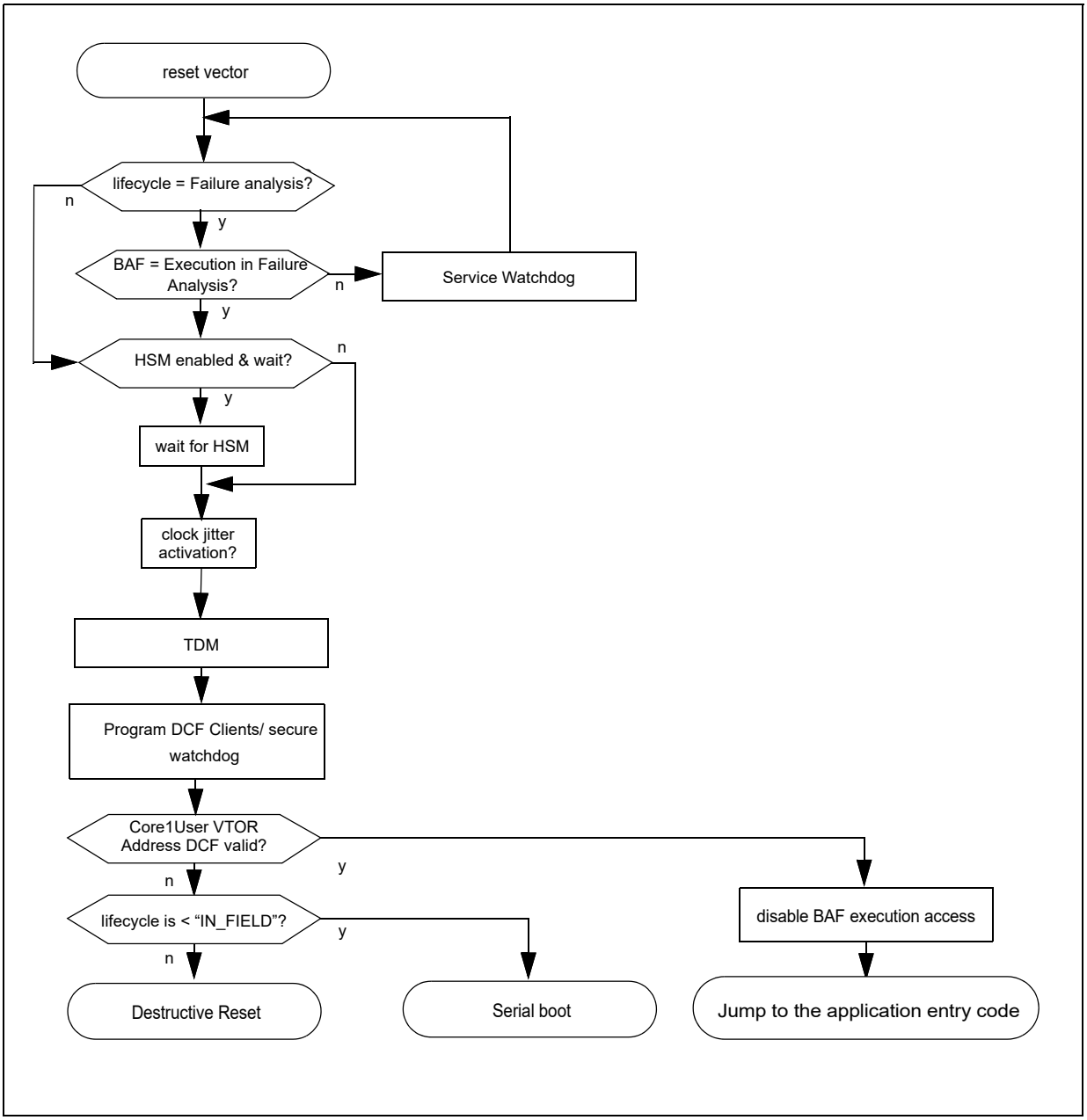
The BAF performs a minimal initialization of core registers and clocks required to enter the above described boot modes, then attempts to interact with the Hardware Security Module (HSM) in the following way:

- HSM: if the HSM is enabled and the “wait for HSM” flag is present, then the boot core waits for the HSM to become ready.
- BAF sets up the exception vector table to catch machine check exceptions, which may occur if there are any uncorrectable ECC errors. In the event of an uncorrectable ECC error, the BAF issues a destructive reset from the machine check exception handler if the exception is due to the reading of instructions from the BAF region. If an ECC error occurs during reading of D-MEM, the BAF still issues a destructive reset. BAF programs the Reset and Clock Control module to issue a destructive reset.
- BAF uses DMEM memory area of the boot CPU as in *Table 253: BAF memory organization* for its stack, data and any code that executes from volatile memory. During the BAF start-up procedure, it initializes this section of memory so that there is no ECC error when BAF accesses this memory area.
- The memory area of the BAF is protected from execution for functional safety considerations. This is achieved by disabling BAF execution in PFLASH control register3 (PFLASHC_PFCR3[BAF_DIS]). At this point the BAF has finished its operation.

18.3.3 Flow of control

The figure below shows the overall flow of control in the BAF. The individual steps are explained in the following sections.

Figure 182. Flow of control



18.3.4 Optionally wait for HSM

If the HSM is enabled (SSCM_UOPS[HSE] = 1), the boot core waits for the HSM to become ready (SSCM_UOPS[HSB = 1]). The core then polls the HSM until a ready flag is set. Both SSCM registers can be set by a DCF record. The boot core polls the HSM to HOST STATUS register (HSM_HSM2HTS) until its least significant bit is set. The boot core also services the watchdog while polling the HSM_HSM2HTS register. This is just in case the HSM is not ready in 16 ms, servicing the watchdog prevents the system from going to reset.

18.3.5 Initialization of BAF DCF clients

18.3.5.1 'Soft' DCF clients

DCF clients are 32-bit hardware registers that are written by the SSCM during reset with data that is stored in DCF record format in UTEST region of flash memory. These DCF records are used to configure module registers, and select device operating modes.

The BAF extends the DCF record concept by implementing soft DCF clients. The 64-bit DCF records include:

- 32 bits of data
- 15-bit chip select field (selects the module which implements the DCF clients)
- 15-bit address field (selects a specific DCF client within the selected module)
- Parity bit (not used with UTEST DCF records)
- Stop bit (to indicate the end of DCF record has been reached)

As part of its execution flow, BAF searches through the list of DCF records in UTEST flash to determine if any of the records are to be processed by BAF.

The search starts at one of two addresses: either the start address of DCF records in UTEST (DCF Start Record), or the address specified in UTEST (Soft DCF Record Start Address). If either of the two least significant bits of the 32-bit address stored at the Soft DCF Record Start Address is set, the default start address at the DCF Start Record is used. If the two least significant bits are clear, this 32-bit value is assumed to be the start address of the search.

The default setting of the two least significant bits is 11b (erased value of flash memory is 0xFFFF_FFFF). A user may intentionally program a dummy address with the least two significant bits set in order to disable the search for an alternate address (as the UTEST block is OTP). Valid search start address must start on a 4-byte boundary.

The DCF record search finishes when a DCF record is read with a STOP bit set. The STOP bit may be set because that memory location is not programmed yet (so the list can be added to) or it can be intentionally programmed STOP record to indicate that no more records are added to the list. BAF does not execute the DCF record with STOP bit set. BAF parses the STOP bit before parsing the rest of the records.

BAF uses Chip Select 14 (CS14) for BAF DCF clients. For records with CS14 asserted, the BAF uses the address field (address[14:0]) in DCF records to determine how to process the 32-bit of data. The soft DCF clients programmed by BAF is listed in *Table 257*.

A group of four Soft DCF clients is used to provide a mechanism to write configuration information to address locations. The first DCF client receive 32-bit address information. The next three DCF Clients receive the data that is to be written to the address contained in the first client. The three data clients allow 32-, 16- and 8-bit data to be written to the address in the first client. These four clients can be accessed repeatedly to write the whole string of data to memory.

Another soft DCF client as stated in *Table 257* provides a simple callback feature. The soft DCF record data contains the start address of a subroutine that the BAF calls before it starts checking for a valid User VTOR Address and exits to customer application code. The callback routine can reside within the BAF or UTEST blocks only. The callback DCF client can be accessed repeatedly in order to execute a callback sequence.

The serial boot DCF client provides a mechanism whose application can inform BAF to perform serial boot or boot from internal flash. Details can be viewed in *Section 18.3.5.5: SER_BOOT_CBACK – BAF serial boot DCF client*.

Table 257. BAF DCF client list

Address (hex)	Name	Label	Description
0x0000–0x01FF	Reserved		
0x0200	Address	ADDR	The address to which DATAX is written
0x0201	32-bit Data	DATA32	The 32-bit data that is written to ADDR
0x0202	16-bit Data	DATA16	The 16-bit data that is written to ADDR
0x0203	8-bit Data	DATA8	The 8-bit data that is written to ADDR
0x0204–0x020F	Reserved		
0x0210	Callback Address	CALLBACK	The address of a callback function
0x0211	Serial boot Callback Address	SER_BOOT_CBACK	The address of serial boot callback function
0x0212–0x0213	Reserved		Reserved for future use.
0x0214	Reserved		Reserved for future use.
0x0215–0x7FFF	Reserved		Reserved for future use.

Note: BAF boots with a watchdog timeout window set to 16ms. Parsing more and more DCF Records may increase boot time. But it must be taken care that number of DCF records must not be very high as it can cause watchdog window timeout.

18.3.5.2 BAF DCF client - Address

BAF parses the DCF records. If BAF finds a DCF record with CS14 set and Address field as 0x0200 while parsing, BAF programs this DCF client. This client is written with a 32-bit address from the Data section of DCF record. This 32-bit address is used in conjunction with any one of the DATA32, DATA16 or DATA8 DCF Data clients. BAF does not immediately use the ADDRESS information, but it is stored and used later in conjunction with a DATAX DCF client (DATA32, DATA16 or DATA8). Then BAF writes the data to the address in memory specified by this ADDRESS DCF client.

The user can create a series of data writes to memory by programming a series of DCF records consisting of pairs of records. The first DCF record of the pair writes to ADDR to provide the address. The second DCF record of the pair writes to the DATA client (DATAX) to provide the data. Subsequent pairs of DCF records can be programmed to write to more memory locations. The number of pairs of DCF records that can be programmed is only limited by the amount UTEST space set aside for DCF records.

DCF Client Address 0x0200

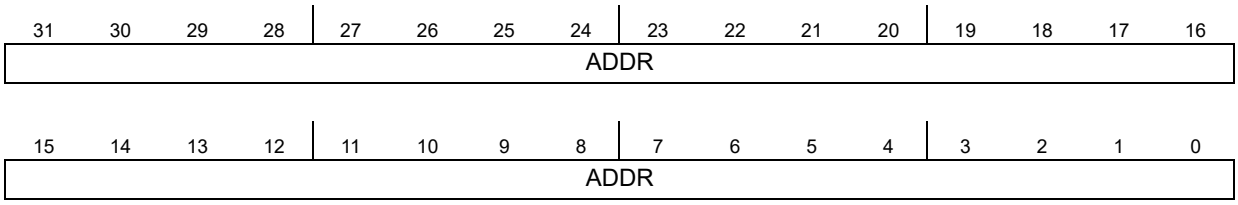


Figure 183. BAF DCF ADDR client configuration

Table 258. BAF DCF ADDR client description

Field	Description
31:0 ADDR	32-Bit Address DCF client This ADDRESS value is used in conjunction with a DATAX value. The BAF writes DATAX value to ADDRESS value in the memory space of the MCU.

18.3.5.3 BAF DCF client - DATAX

Caution: While using DATAX DCF records, it must be taken in account that BAF does not take care of ECC scheme and can generate ECC, if write is performed in memory locations which are protected by ECC and are not pre-initialized as required.

18.3.5.3.1 DATA32 – 32-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0201 while parsing, BAF programs this 32-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant 2 bits to ‘0’ (Only 32-bit aligned writes are supported) and writes this 32-bit data to the address.

DCF Client Address 0x0201

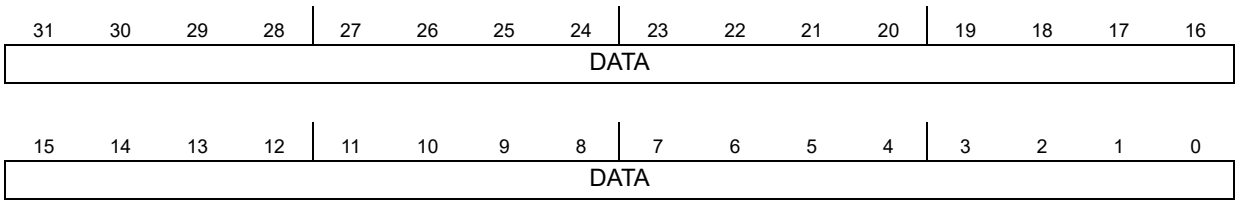


Figure 184. DATA32 DCF client configuration

Table 259. DATA32 DCF client configuration

Field	Description
31:0 DATA	32-Bit DCF data

18.3.5.3.2 DATA16 – 16-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0202 while parsing, BAF programs this 16-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant bits to ‘0’ (Only 16-bit aligned

writes are supported) and writes the least significant 16-bit data to the address. Data in the 16 most significant bits are ignored.

DCF Client Address 0x0202

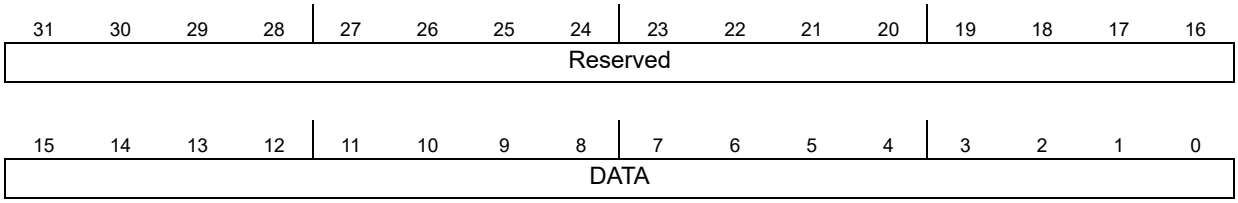


Figure 185. DATA16 DCF Client Configuration

Table 260. DATA16 DCF client configuration

Field	Description
15:0 DATA	16-Bit DCF data

18.3.5.3.3 DATA8 – 8-bit DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0203 while parsing, BAF programs this 8-bit DCF client. BAF takes the memory address provided by the ADDR client, and writes the least significant 8-bit data to the address. Data in the 24 most significant bits are ignored.

DCF Client Address 0x0203

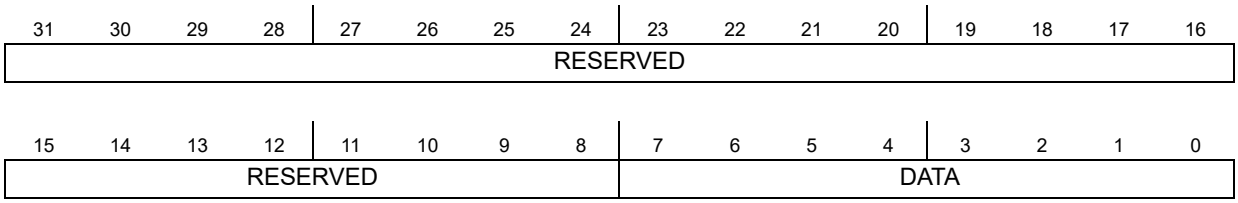


Figure 186. DATA8 DCF Client Configuration

Table 261. DATA8 DCF client configuration

Field	Description
7:0 DATA	8-bit DCF data

18.3.5.4 CALLBACK – BAF callback DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0210 while parsing, BAF programs this callback DCF client. This client is written with a 32-bit start address of a function which is invoked by BAF. The BAF code checks the start address is within the UTEST or within the BAF flash memory blocks, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory blocks, no function call is executed. The callback is aborted and BAF moves on to the next DCF records. The function that is called is expected to comply with Embedded Application Binary Interface (EABI) for Arm® found in Arm® Architecture web site and to

terminate in an orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function and no return value is expected.

Caution: Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

BAF executes the callback function as the DCF records are parsed. Therefore, several callback functions can be executed by writing several DCF records that write to the callback DCF clients. The callback functions are executed in the same order as the DCF records.

18.3.5.5 SER_BOOT_CBACK – BAF serial boot DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0211 while parsing, BAF programs this serial boot DCF client. This DCF client is written with a 32 bit start address of function which is invoked by BAF. The BAF code checks the start address is within UTEST or within the BAF flash memory block, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory block, no function call is executed. The DCF client execution is aborted and BAF moves on to the next DCF record. The function that is called is expected to comply with Embedded Application Binary Interface (EABI) for Arm® found in over Architecture web site and to terminate in orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function.

Warning: No parameters are passed to the function invoked by the BAF. Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

The serial boot callback function returns a char value. The returned value informs BAF if serial boot or normal internal flash boot must be executed (refer to *Table 262* for details regarding return value).

Table 262. Serial boot callback function return value definition

Return values	Description
0x00	Boot from internal flash using standard search for boot record.
0x01	Serial boot using both M_CAN and UART (in UART mode). **Note:** Serial boot is conditioned by life cycle as described in *Figure 182*.

18.3.6 Production disable activation protocol and implementation

Module failure analysis is a standard requirement. But some OEMs have an additional requirement that upon entering failure analysis mode, the MCU operation is modified in such a way that the MCU can never be used in production ECU. The chosen operational modification for production disable is to apply clock jitters to can modules so that it prevents active communications with another can device on a can network. For detail regarding Production Disable refer to the Security manual.

The BAF maintains a 64-bit flag in BAF code flash region as explained in *Section 18.2.1.2: Clock jitter activation constant*. On every POR BAF copies the clock jitter activation constant from this flag to clock jitter enable register in PASS module to enable jitters on can baud clock. Default value of this flag is shown in *Section 18.2.1.2: Clock jitter activation constant*. Hence, jitter on the MCAN baud clock is disabled if the default value is used. To enable jitter on MCAN baud clock, the value of this flag must be modified to 0xFFFF_FFFE.

The BAF flash memory is write protected and locked.

18.3.7 TDM diary operation

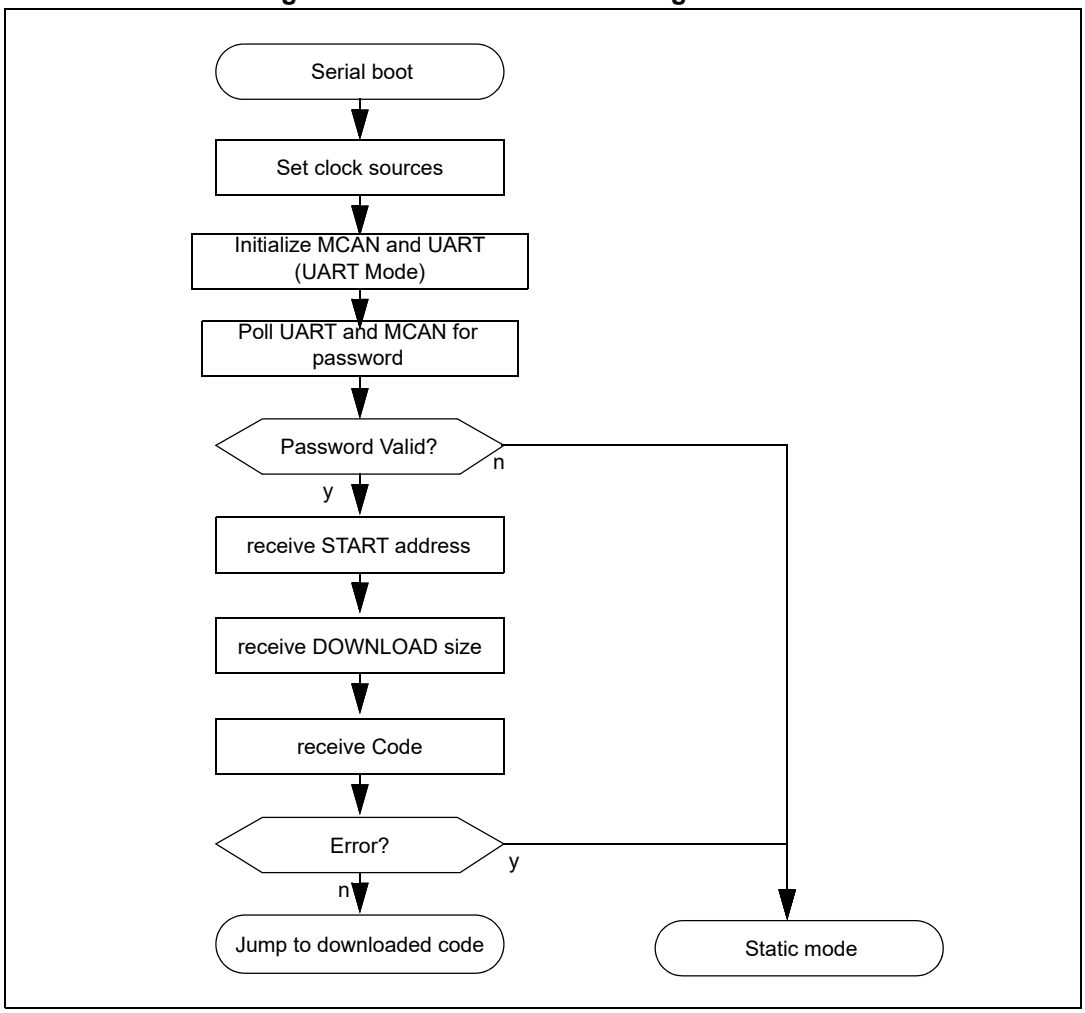
On every reset, BAF copies the first four bytes of last record of every Tamper Detect Region to the corresponding Software Tamper Override Key Region Registers in TDM. BAF reads the base address of TDM Diary from DBA register in TDM. For more details on TDM refer to the device Security Manual. This is done as a part of TDM Diary Override enhancement.

18.3.8 Optionally perform a serial boot

If no Core1 User VTOR Address DCF record is found in one of the locations mentioned above, then BAF determines the Life Cycle status of the device. If the Life Cycle is lower than In Field, it attempts a serial boot. Otherwise, the boot has failed and the BAF issues a destructive reset.

If the customer selects serial boot using serial boot using serial boot DCF client, then BAF determines the Life Cycle status of the device. If the Life Cycle is lower than Failure Analysis, it attempts a serial boot.

Figure 187. Flow of control during serial boot



The clock source in the RCC is configured (with IRCOSC for UART and with XOSC for CAN) and the PADs are set up for UART and M_CAN signals, then the code is downloaded using UART or M_CAN in Serial Boot mode. In the absence of errors causing the device to be placed in Static mode, the device state is restored to resemble the out-of-reset state and the downloaded code is executed.

Serial boot allows the download of application to internal SRAM via UART or M_CAN interface. After downloading the application image to SRAM, the BAF jumps to the start address provided as part of download protocol mentioned below and executes the downloaded application.

At the start of the process both UART and M_CAN are configured in passive listening mode. As soon as either interface receives a valid message, the receiving interface is configured in full Tx and Rx mode. The non receiving interface is switched back to reset state and does not participate further in serial boot.

If a CAN message of 8 bytes with ID 0x11 is received on the M_CAN controller first, the BAF program:

- Disables UART.
- Configures UART RXD input mux back to reset state.
- Transitions to CAN serial download protocol routine.

If a CAN message not containing 8 bytes or NOT with ID 0x11 is received by M_CAN controller or a M_CAN controller generates a CAN error, it is probable that the tool is sending a UART frame. Under such circumstances BAF program:

- Ignores the CAN message, if a message arrived.
- Clears any pending CAN error.
- Continues monitoring for valid CAN messages.

If any byte from UART is received first, the BAF program:

- Checks if the first byte is 0xFE of the password. If not it continues polling. The data may be a valid CAN message.
- If it is first byte of the password (0xFE), then it disables the M_CAN controller and restores to its reset state.
- Configures the M_CAN pads to their reset state.
- Transitions to UART serial boot (UART mode).

18.3.8.1 Serial boot mode protocol

From a high-level perspective, the download protocol over M_CAN and LIN interfaces follows the steps below:

1. Host transmits the 64-bit password to MCU.
2. Host transmits start address, size of downloaded code in bytes, NO_ECHO bit to MCU.

   NO_ECHO bit signifies if actual download data to be echoed or not. If NO_ECHO bit is set, then data is not echoed otherwise if NO_ECHO bit is clear, then data is echoed.
3. Host transmits a sequence of data bytes that are to be executed.
4. The boot CPU executes code from start address provided in step 2.

Each step must be completed before the next step starts.

The exact protocol between UART and M_CAN is slightly different.

18.3.8.2 UART download protocol

The communication is performed in half-duplex mode, any transmission from the host is followed by the MCU transmission.

1. The host sends data to the MCU and starts waiting.
2. The MCU echoes the received 8-byte of password, 4-byte of load address and 4-byte of NO_ECHO bit and 31-bit of download size to the host. The MCU echoes actual download data depending upon NO_ECHO bit.
3. The host verifies whether the echo is correct.
   - If the data is correct, the host can send the next byte of data.
   - If data is not correct, it is assumed the MCU has not correctly received the data and the only practical option is for the Host to reset the MCU and start again.

A more detailed description of these steps follows.

All multi-byte data structures are sent with MSB first.

The tool sends data in the following order:

1. 8 bytes that represent the password.
2. 4 bytes that represent the start address where the code is to be downloaded.
3. 4 bytes that represent the number of bytes to be downloaded, NO_ECHO bit.
4. *x* bytes of code that is downloaded to SRAM and executed (*x* must match the value specified in step *3.*).

18.3.8.3 M_CAN download protocol

The M_CAN download protocol uses very similar structure to the UART protocol, but it makes use of the message ID's provided by M_CAN.

The tool sends data packed into standard CAN message in the following manner:

1. A message with 0x11 as the ID and an 8-byte length to send the password. The MCU transmits the same message, but with an ID of 0x1.
2. A message with 0x12 as the ID and 8-byte length to send the start address, size of the code.The MCU transmits back the same data but with an ID of 0x2.
3. A message with 0x13 as the ID is used to send data that is to be downloaded to SRAM. The MCU transmits the same data with an ID of 0x3.

18.3.8.4 Download 64-bit password and password check

The first 64 bits received represent the password. The MCU only supports public passwords. The received password is always compared with the public password 0xFEED_FACE_CAFE_BEEF.

- If the correct password is received, BAF continues its task.
- If an incorrect password is supplied, BAF puts the device into static mode (low power safe mode).
- If 64 bits of password data is not received after approximately 30s, BAF causes a destructive reset of the MCU.

18.3.8.5 Download start address, NO_ECHO bit and code size

After the password, the next 8 bytes received by the MCU contains a 32-bit start address, NO_ECHO bit and 31-bit code length as shown in *Figure 188* and *Figure 189*.

The NO_ECHO bit is used to indicate if download data must be echoed by the MCU. If the NO_ECHO bit is set (for example, NO_ECHO = 1), then only 8 bytes of password, 4 bytes of download address, and 4 bytes of download size are echoed, but download data is not echoed. If the NO_ECHO bit is clear, then every byte sent to the MCU must be echoed.

The start address defines where the received data is stored and where the MCU branches after the download is finished. The three LSB bits of the start address are ignored by BAF, such that the load address is 64-bit double-word aligned.

The length determines how many data bytes have to be loaded.

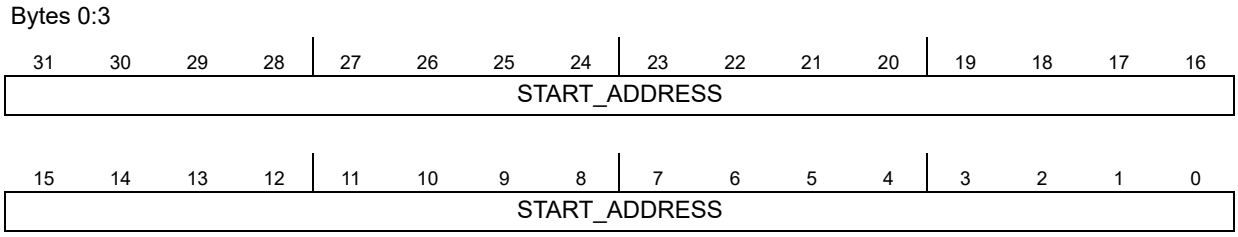


Figure 188. Start address

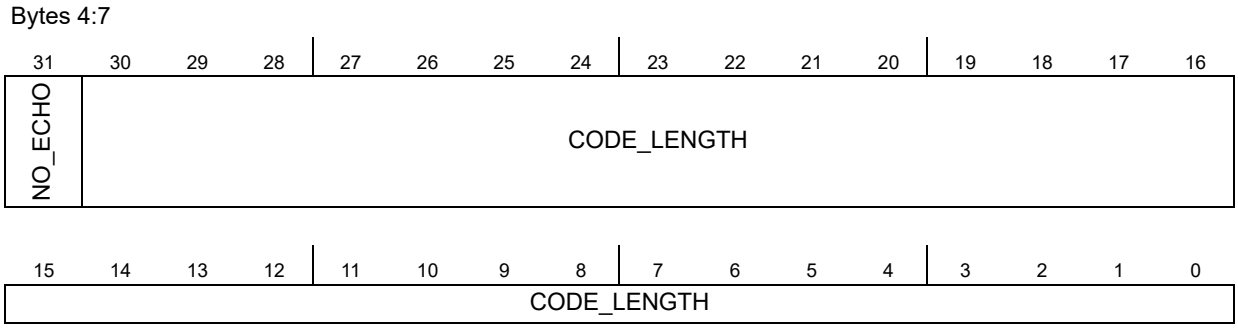


Figure 189. NO_ECHO bit and code size

18.3.8.6 Download data

Each byte of data received is downloaded into device SRAM starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until a number of bytes of data received matches the code length specified in the previous protocol step.

Since the SRAM is protected by 64-bit wide Error Correction Code (ECC), BAF always writes to SRAM grouped in 64-bit double-words. If the last byte does not fall into a 64-Bit bounder, the BAF fills it with zero's and renders the data aligned on a 64-bit bounder. After the last data is written to SRAM, the BAF writes a dummy double-word (0x0000_0000_0000_0000) to the next address location. This is written to avoid possible ECC error during core prefetch.

18.3.8.7 Execute code

The BAF program waits for the last echo message transmission to be completed. Then it restores the initial MCU configuration and jumps to the loaded code at Start Address which was received in step 2 of the protocol. At this point the BAF has finished its tasks and the MCU is controlled by new code executing from SRAM.

18.3.9 Serial boot configuration

18.3.9.1 UART configuration

Boot using UART protocol is implemented by UART module.

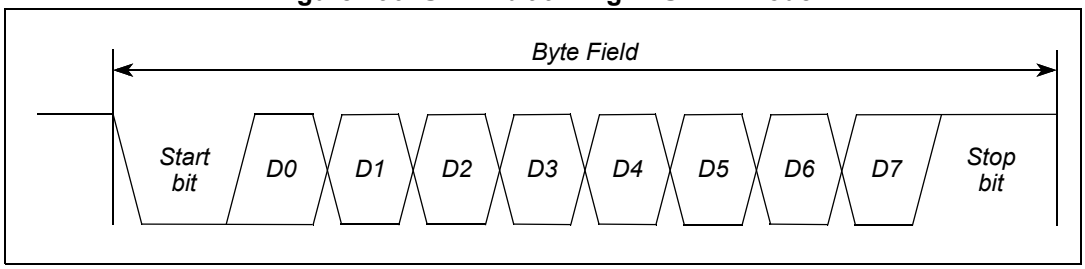
Note: Refer to the pin configuration table in Chapter 5: Device configuration for pins used by the UART module.

When Serial Boot mode is started the UART_Rx is configured as an input and multiplexed to the UART module. The UART_Tx remains in a passive input state until the first byte of protocol is received. It is then configured as an output and multiplexed to UART module. The application may not be using the Serial Boot Mode function, and may be using the pin used by UART_Tx as an input, which may have some external hardware driving the pin. Inadvertently driving this pin as an output would cause contention and possible damage.

The UART module operates in half-duplex mode. The interface is either transmitting a byte or receiving a byte, but never both at the same time. It is expected that UART pins are connected to a physical layer bus driver/receiver (PHY). Many PHYs enforce half-duplex mode by using a single data channel for both transmitting and receiving data. The side effect of this is that all data transmitted on Tx pin is also received on Rx pin. To prevent the UART module receiving its own transmission, the Rx pin is blocked during Tx pin transmission.

The UART controller is configured to operate with an 8-bit data frame without parity bit and 1 stop bit.

Figure 190. UART bit timing in UART mode

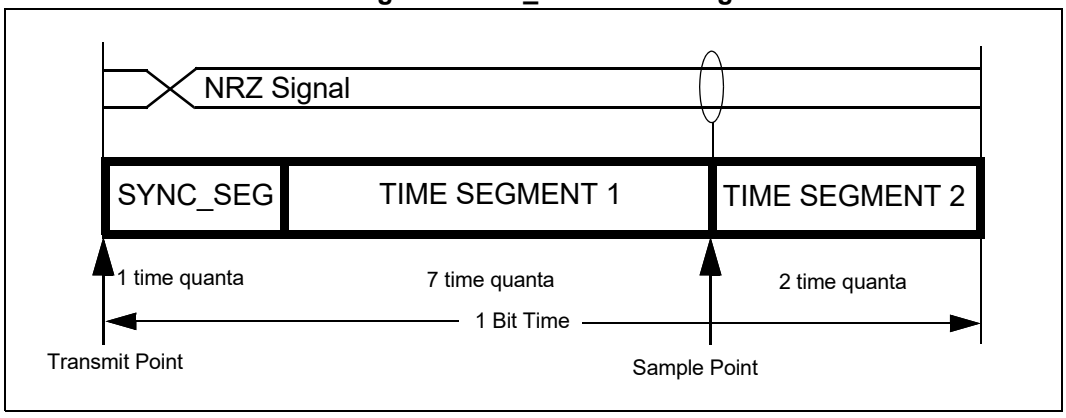


18.3.9.2 M_CAN configuration

The M_CAN controller is configured to operate at a baud rate equal to crystal frequency divided by 40, using the standard 11-bit identifier format detailed in CAN 2.0A specification (refer to *Table 263* for baud rates).

The Bit timing is configured as shown in *Figure 191*.

Figure 191. M_CAN Bit Timing



18.3.9.3 Baud rates

The UART baud rate is set at 250kbaud. *Table 263* below shows some possible baud rates.

The M_CAN baud rate is set by the frequency of the external crystal/40.

Table 263. M_CAN baud rates

Frequency	M_CAN baud rate
8 MHz	200 Kbaud
12 MHz	300 Kbaud
16 MHz	400 Kbaud
20 MHz	500 Kbaud
40 MHz	1000 Kbaud

18.3.9.4 UART and M_CAN pin configuration

Refer to *Chapter 5: Device configuration* for pin details.

18.3.9.5 Protocol summary

Table 264 summarizes the UART protocol and BAF action during this boot mode.

Table 264. UART serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	64-bit password (MSB first)	64-bit password (MSB first)	Password checked for validity and compared with stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	NO_ECHO bit + 31 bits of download data size in bytes (MSB first)	NO_ECHO bit + 31 bits of download data size in bytes (MSB first)	Size of download is stored for future use.
4	8 bits of raw binary data	If NO_ECHO bit is set, data is not echoed, otherwise data is echoed.	8 x 8 bits of data are packed into 64-bit double-words. These double-words are stored in SRAM starting from the Load address. The Load address increments until the number of data received and stored matches the size as specified in the previous steps.
5	None	None	Branch to downloaded code

The table below shows the M_CAN serial boot protocol.

Table 265. M_CAN serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	CAN ID 0x011 + 64-bit password	CAN ID 0x001 + 64-bit password	Password checked against fixed password (0xFEED_FACE_CAFE_BEEF). Watchdog timer is refreshed if the password check is successful.
2	CAN ID 0x012 + 32-bit store address + 32-bit (1 NO_ECHO bit and 31 of download data size in bytes)	CAN ID 0x002 + 32-bit store address + 32-bit (1 NO_ECHO bit and 31 of download data size in bytes)	Load address and size of download are stored for future use.
3	CAN ID 0x013 + 8 to 64-bits of raw binary data	If NO_ECHO is set, data is not echoed, otherwise CAN ID 0x003 + data.	Each byte of data received is store in MCU memory, starting at the address specified in the previous step and incrementing until the amount of data received and stored, matched the size as specified in the previous step.
4	None	None	The BAF returns IO pins and CAN module to their reset state, then branches to the first address the data was stored to (As specified in step 2).
5	None	None	Branch to downloaded code.

18.4 Resources accessed by BAF code execution

The BAF code execution uses several IPs to achieve the main functions.

- Reset and clock control (RCC)
  Several clock configurations are made depending on the boot mode of the BAF. All settings are restored before execution of the application code (in flash memory or received via serial download).
- UART and M_CAN
  The UART and M_CAN modules are only used if a serial download boot is attempted, or from the SER_BOOT_CBACK callback function. All settings are restored before the application code in system RAM is executed.
- PFLASH
  PFLASH module is used to disable BAF execution. The two PFLASH are used.
- Interrupts
  No interrupts/exceptions are used or enabled, except the Machine Check exception, to handle ECC errors.
- TDM
  TDM for implementing TDM Diary enhancements.
- SSCM
  Used to read some register bits (mirrors of DCF).
- PASS
  Used for clock jitter enabling.

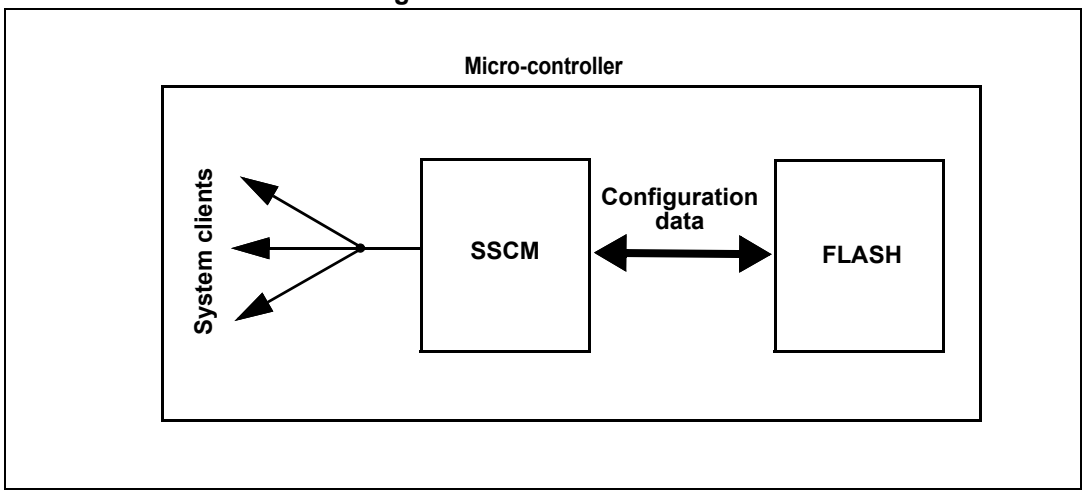
19 System status and configuration module (SSCM)

19.1 Introduction

19.1.1 Overview

The System Status and Configuration Module (SSCM) is aimed at reading the system configuration from the Flash and distributing to various clients inside the micro-controller, pictured in the figure below.

Figure 192. SSCM overview



19.1.2 Features

The SSCM includes these distinct features:

- System Configuration and Status
  - Device Mode and System Status
  - Decodes the Device Life Cycle

19.1.3 Modes of operation

The SSCM operates identically in all system modes.

19.2 Memory map and registers definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

Table 266 shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

Note: All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit READ/WRITE to address Base + 0x0002, but performing a 16-bit access to Base + 0x0003 is illegal.

Table 266. SSCM memory map

Address offset	Registers	Access	Reset value	Section
0x0000	SSCM System Status (STATUS)	R/W	0xyyy0(1)	*Section 19.2.1.1*
0x0004	SSCM Error Configuration Register (ERROR)	R/W	0x0000	*Section 19.2.1.2*
0x0008–0x0009	Reserved(2)			
0x000A–0x001F	Reserved			
0x0020	SSCM HSM and User Option Status Register (UOPS)	R	0xyyyy_yyyy(1)	*Section 19.2.1.3*
0x0024–0x0033	Reserved			
0x0034	Life Cycle Status Register (LCSTAT)	R	0x0000_0y0y(1)	*Section 19.2.1.4*
0x0038	OTA Signature Address Location register (OSAL)	R	0xyyyy_yyyy(1)	*Section 19.2.1.5*
0x003C–0x0048	Reserved			
0x004C	Non Active Context Logical Address register (NACLA)	R	0xyyyy_yyyy(1)	*Section 19.2.1.6*

1. Refer to register description for reset value details.
2. No register abort on this location.

19.2.1 Registers descriptions

The following registers are available in the SSCM. Those bits that are shaded out are reserved for future use. To optimize future compatibility, these bits must be masked out during any read/write operations to avoid conflict with future revisions.

19.2.1.1 System status register (STATUS)

The System Status register reflects the current state and the memory configuration of the system. It also contains the JTAG ID.

Offset 0x0000 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	CER	0	0	0	1	0	0	0	1	1	0	0	0	0	0
W		w1c														
Reset	0	0	0	-(1)	-(1)	1	0	0	0	1	1	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	JPIN										1	MREV				1
W																
Reset	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	1	0/1(2)	0/1(2)	0/1(2)	0/1(2)	1

Figure 193. System Status Register (STATUS)

1. Reset value depends on the device status after leaving reset.
2. Reset value reflects the JTAG ID of the device.

Table 267. STATUS field descriptions

Field	Description
30 CER	Configuration Error This field indicates that the SSCM has detected a configuration error while loading DCF clients (with Parity enabled). This error may have been detected during the reset sequence. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct If a non-permanent (transient) error is detected, this bit can be cleared by writing a 1. If the error is permanent, the bit reappears immediately.
15:6 JPIN	JTAG Part ID Number
4:1 MREV	Minor Mask Revision

19.2.1.2 Error configuration register (ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Offset: 0x0004 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	OTA_ER	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is unused.

Figure 194. Error Configuration Register (ERROR)

Table 268. ERROR field descriptions

Field	Description
2 OTA_ER	This bit captures the error scenario if during the functional reset phase (or when BIST is disabled) the SSCM OTA reading of the signature slot and slot next to signature slot are inconsistent with what is read during the destructive reset phase. This bit is only valid for software when OTA is enabled. This bit is cleared by functional reset.

19.2.1.3 HSM and user option status register (UOPS)

Offset: 0x0020 Access: Read Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	IWDG2_HW_EN	IWDG1_HW_EN	0	HSM_EN_FA	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0/1(1)	0/1(1)	0	0/1(1)	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BAF_FA	0	0	0	0	0	FTMPE	FTMPL	0	0	0	0	HSB			HSE
W																
Reset	0/1(1)	0	0	0	0	0/1(1)	0/1(1)	0/1(1)	0	0	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. Values are determined by DCF record and life cycle. Default is 0.

Figure 195. HSM and User Option Status Register (UOPS)

Table 269. UOPS field descriptions

Field	Description
29 IWDG2_HW_EN	Independent Watchdog 2 HW Enable 1 IWDG2 is enabled by HW after reset. 0 IWDG2 is not enabled by HW. SW must enable it. This field can be set via DCF record.
28 IWDG1_HW_EN	Independent Watchdog 1 HW Enable 1 IWDG1 is enabled by HW after reset. 0 IWDG1 is not enabled by HW. SW must enable it. This field can be set via DCF record.
26 HSM_EN_FA	HSM Enable in Failure Analysis 0 HSM not enabled in FA 1 HSM enabled in FA This field can be set via DCF record (refer to the SR5E1x Microcontroller Security Reference Manual for the HSM details).
15 BAF_FA	BAF execution mode when bypass mode DCF is zero and lifecycle is FA 0 BAF enters an infinite loop 1 BAF executes as normal This field can be set via DCF record (refer to the SR5E1x Microcontroller Security Reference Manual for the HSM details).

Table 269. UOPS field descriptions (continued)

Field	Description
9 FTMPE	Flash Test Mode Protection Enable Enable Flash Test Mode Protection during Failure Analysis 0 Flash Test Mode Protection during Failure Analysis is disabled 1 Flash Test Mode Protection during Failure Analysis is enabled This field can be set via DCF record (refer to the SR5E1x Microcontroller Security Reference Manual for the HSM details).
8 FTMPL	Flash Test Mode Protection Lock during Failure Analysis. It is used by software to determine that the related DCF client has been written by a DCF record and locked, since it is a 'write-once' DCF. The once set, the value of the related DCF cannot be changed anymore. This field can be set via DCF record (refer to the SR5E1x Microcontroller Security Reference Manual for the HSM details).
3:1 HSB	HSM Boot Configuration 001 When the device is in BAF Code Boot mode, BAF waits for HSM ready flag before allowing application code to start. In User Code Boot mode the SSCM waits for HSM ready flag before allowing application code to start. Others: When the device is in BAF Code Boot mode, BAF does not wait for HSM ready flag before allowing application code to start. In User Code Boot mode the SSCM does not wait for HSM ready flag before allowing application code to start. This field can be set via DCF record (refer to the SR5E1x Microcontroller Security Reference Manual for the HSM details).
0 HSE	HSM Enabled This bit is set to 1 if HSM is enabled via DCF record.

19.2.1.4 Life cycle status register (LCSTAT)

Offset 0x0034 Access: Read Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	LC		
W																
Reset	0	0	0	0	0	0	0/1(1)	0/1(1)	0	0	0	0	0	0/1(2)	0/1(2)	0/1(2)

1. Reset value reflects the Test Life Cycle of the device.
2. Reset value is the Life Cycle of the device.

Figure 196. Life Cycle Status Register (LCSTAT)

Table 270. LCSTAT field descriptions

Field	Description
2:0 LC	Life Cycle As determined by the SSCM by reading the LC slots. The encoding is: 000 FAIL_ANALYSIS 001 Reserved 010 PROD_OEM 011 CUST_DELIV 100 Reserved 101 Reserved 110 ST_PROD 111 IN_FIELD

19.2.1.5 OTA signature address location register (OSAL)

This register holds the address of the empty slot in OTA signature block.

The software must write the new signature in this slot once the OTA flash update is completed. When the OTA block is full the OSAL[31:0] = 0xFFFFFFFC.

These bits are only meaningful if OTA is enabled.

Offset 0x0038 Access: Read Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OTA_SIG_ADDR															
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OTA_SIG_ADDR														0	0
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0	0

1. Reset value depends on the DCF value.

Figure 197. OTA Signature Address Location register (OSAL)

Table 271. OSAL field descriptions

Field	Description
31:2 OTA_SIG_ADDR	This field represents the OTA signature block address pointer to update with a signature.

19.2.1.6 Non active context logical address register (NACLA)

The register reflects the logical address of Non active context. The address of this register is only useful for software when the OTA is enabled by DCF.

Offset 0x004C Access: Read Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NON_ACTIVE_LOGICAL_ADDR															
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NON_ACTIVE_LOGICAL_ADDR															
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. Reset value is device specific.

Figure 198. Non Active Context Logical Address register (NACLA)

Table 272. NACLA field descriptions

Field	Description
31:0 NON_ACTIVE_ LOGICAL_ADDR	Non active logical address which has to be used for firmware update location.

19.3 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

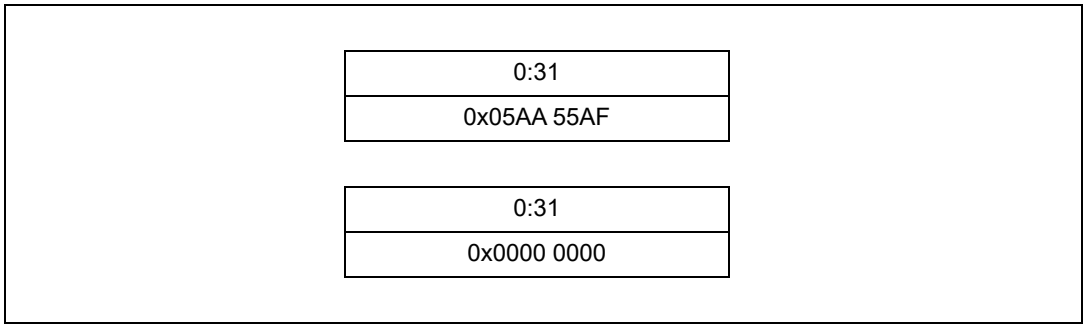
19.3.1 DCF mechanism

The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET the user can store a series of DCF records - the device processes these records during the system reset sequence before the CPU leaves reset. The first record needs to be a start record followed by an application-specific number of data records as required, and finally a stop record.

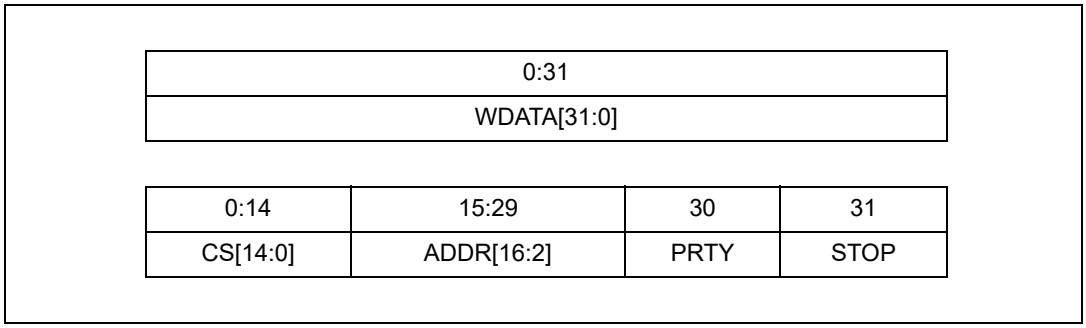
Each record is 64 bits in length. *Figure 199* shows the DCF start record. The start record must be placed at UTEST_OFFSET to indicate to the device that the following data records must be processed.

Figure 199. DCF start record



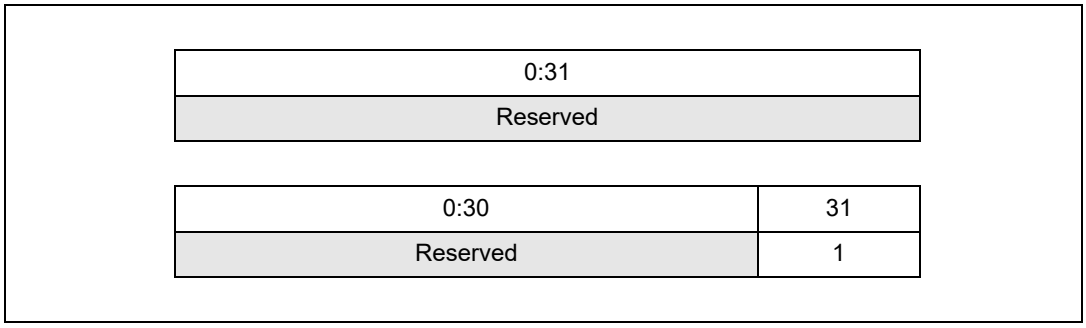
The data records are structured similar to a CPU write instruction - 15 bits of client select, 15 bits of address, 32 bits of payload data plus a STOP bit - refer to *Figure 200*. In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Figure 200. Format of a DCF data record



The stop record indicates that processing must stop. The general format of the stop record is shown in *Figure 201*. Only the STOP bit needs to be 1 in order to form a stop record – All other bits are ignored. An unprogrammed location in UTEST flash is interpreted as a stop record.

Figure 201. DCF stop record



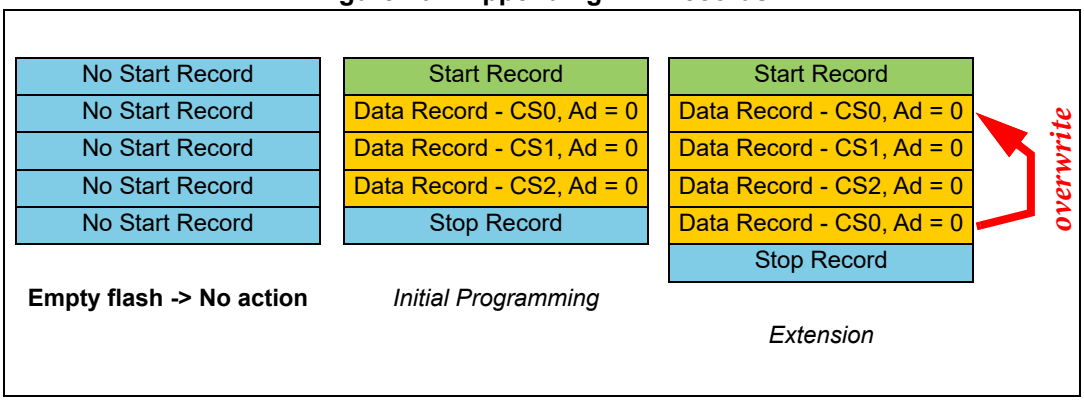
If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 273*.

Table 273. Series of DCF records in UTEST

ADDR offset	DATA			
0x00	0x05AA_55AF			
0x04	0x0000_0000			STOP=0
0x08	WDATA[31:0]			
0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
0x10	WDATA[31:0]			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
8n - 1 + 0x0	Reserved			
8n - 1 + 0x4	Reserved			1
8n + 0x0				
8n + 0x4				

There must never be an unprogrammed record in the data structure, as that would be interpreted as a stop record, so subsequent records would be ignored. This allows programming the records in several sessions, each time appending new records at the end of the list, as shown in *Figure 202*.

Figure 202. Appending DCF records



In this case, a record may overwrite a client's value which was already set by a previous record. However whether such an operation is allowed depends on the rules given for the client - some clients are write-once, only allow setting bits but not clearing, and so on.

The parity bit (PRTY) is required for some DCF records. The parity scheme used is even parity, so the number of 1s in the WDATA and PRTY fields needs to be even. (For example, if the WDATA field has the value 0x0000_0001 then the PRTY field needs to be set to '1', so that the total number of 1s is even).

19.3.2 ECC error monitoring

During flash memory scanning (such as Life Cycle or DCF records), the SSCM monitors the flash memory status signals for ECC and other unexpected access errors. If such an error occurs the SSCM requests a destructive reset.

19.3.3 Life cycle

The SSCM determines the Life Cycle of the device by reading the Life Cycle slots from UTEST flash memory area. The read operation is done during the reset phase with normal timings and it is protected by both operating monitors and ECC check. In addition a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- ST Production
- Customer Delivery
- OEM Production
- In Field
- FA

The LC is written into 5 slots, 256 bits each, at fixed locations in UTEST flash memory block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- The valid field
- The invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 status:

Table 274. Life cycle slots

Use case	Offset	LC slot	Valid field (128 bits)		Invalid field (128 bits)		LC slot status
1	0x00-0x10	1	Erased	Erased	Erased	Erased	Erased
2	0x00-0x10	1	Marked	Erased	Marked	Erased	Inactive
3	0x00-0x10	1	Marked	Erased	Erased	Erased	Active
4	0x00-0x10	1	Any other value				Illegal

“Marked” in this case means that the value has been programmed with the bit pattern 0x55AA_50AF_55AA_50AF, “Erased” is detected by the bit pattern 0xFFFF_FFFF_FFFF_FFFF.

Table 275 shows how the slots are arranged in memory.

Table 275. Life cycle slots in memory

Offset	LC slot word
0x00 - 0x07	Valid[63:0]
0x08 - 0x0F	Don't care

The Life cycle is determined as shown in *Table 276*. The priority of the entries is from top to bottom, so the first row which applies determines the resulting Life cycle.

Table 276. Resulting Life cycle

LC slot 0 ST_PROD(1)	LC slot 1 CUST_DEL(1)	LC slot 2 OEM_PROD(1)	LC slot 3 IN_FIELD(1)	LC slot 4 FA(1)	Resulting Life cycle
Active	Erased	Erased	Erased	Erased	ST Production
Inactive	**Active**	Erased	Erased	Erased	Customer Delivery
Inactive	*Inactive*	**Active**	Erased	Erased	OEM Production
Inactive	*Inactive*	*Inactive*	**Active**	Erased	In Field
Inactive	*Inactive*	*Inactive*	*Inactive*	**Active**	FA
Any other combination				Erased	In Field
Any other combination					System Reset

1. Erased = all slot bits at '1'

19.4 Initialization and application information

19.4.1 Reset

The reset state of each individual bit is shown within *Section 19.2.1: Registers descriptions*.

19.5 Additional safety measures

19.5.1 Spurious reset protection

The SSCM implements protection against spurious module resets (for example, resets which only effect the SSCM but not other modules) by gating the state machines with the expected status of the RCC. In case of a spurious reset the SSCM does not interfere with the flash memory bus or overwrite configuration registers. Status internal to the SSCM are lost, however.

20 General-purpose I/Os (GPIO)

20.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (MODER, OTYPER, OSPEEDR and PUPDR), two 32-bit data registers (IDR and ODR) and a 32-bit set/reset register (BSRR). In addition all GPIOs have a 32-bit locking register (LCKR) and two 32-bit alternate function selection registers (AFRH and AFRL).

20.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (IDR) or peripheral (alternate function input)
- Bit set and reset register (BSRR) for bitwise write access to ODR
- Locking mechanism (LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

20.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the BSRR register is to allow atomic read/modify accesses to any of the ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 203 shows the basic structure of a standard I/O port bit. *Table 277* gives the possible port bit configurations.

Figure 203. Basic structure of an I/O port bit

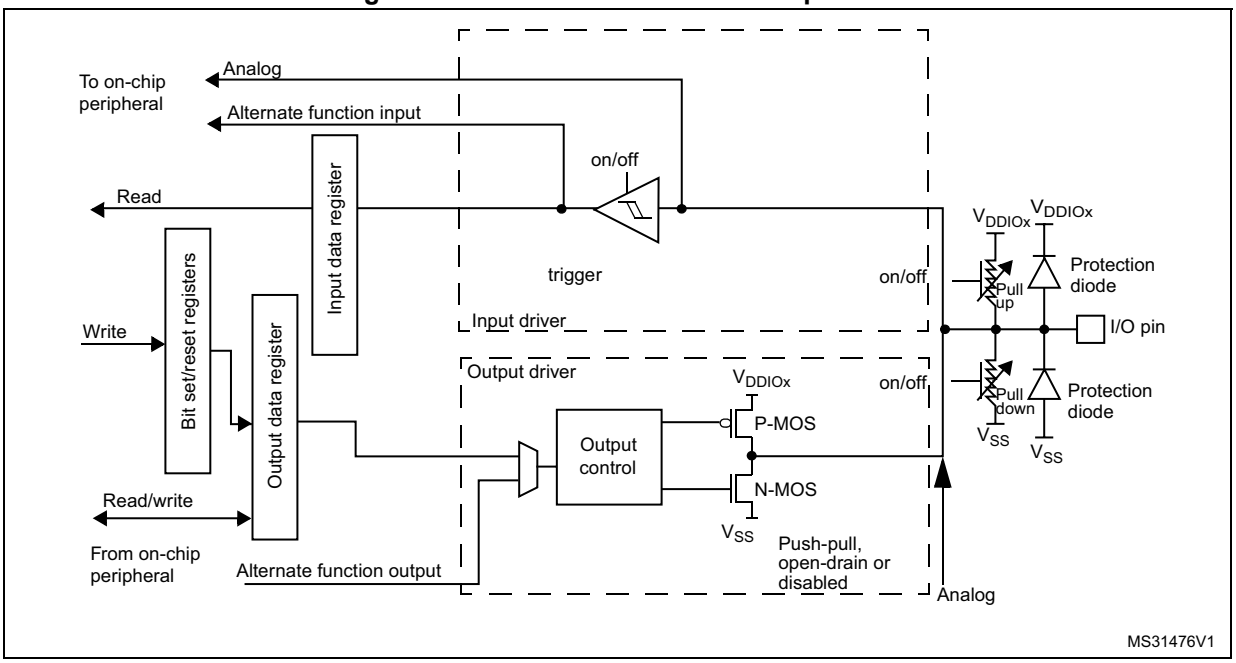


Table 277. Port bit configuration table

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]	PUPD(i) [1:0]		I/O configuration(1)	
01	0	SPEED [1:0]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]	0	0	AF	PP
	0		0	1	AF	PP + PU
	0		1	0	AF	PP + PD
	0		1	1	Reserved	
	1		0	0	AF	OD
	1		0	1	AF	OD + PU
	1		1	0	AF	OD + PD
	1		1	1	Reserved	

Table 277. Port bit configuration table (continued)

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration(1)	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/output	Analog, PD
	x	x	x	1	1	Reserved	

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

20.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are separated in two groups: 3 dedicated pins with no GPIO or AF capability, and 2 configurable pins. After reset they are configured as follows:

- dedicated JTCK/SWCLK in input with pull-down
- dedicated JTMS/SWDIO in input with pull-up
- JCOMP in input with pull-down
- PG3 as alternate function JTDI in input with pull-up
- PG4 as alternate function JTDO in output push-pull low level with no pull-up/down

When the pin is configured as output, the value written to the output data register (ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the PUPDR register.

20.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to sixteen alternate function inputs (AF0 to AF15) that can be configured through the AFRL (for pin 0 to 7) and AFRH (for pin 8 to 15) registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through MODER register.
- The specific alternate function assignments for each pin are detailed in the SR5E1x pinout Microsoft Excel® file attached to the IO_Definition document.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:

- **Debug function:** after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **GPIO:** configure the desired I/O as output, input or analog in the MODER register.
- **Peripheral alternate function:**
  - Connect the I/O to the desired AFx in one of the AFRL or AFRH register.
  - Select the type, pull-up/pull-down and output speed via the OTYPER, PUPDR and OSPEEDR registers, respectively.
  - Configure the desired I/O as an alternate function in the MODER register.
- **Additional functions:**
  - For the ADC, DAC, and COMP, configure the desired I/O in analog mode in the MODER register and configure the required function in the ADC, DAC, and COMP registers.

    As indicated above, for the additional functions (such as DAC), the output is controlled by the corresponding peripheral. Care must be taken to select the I/O port analog function before enabling the additional function output in the peripheral control register.
  - For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the SR5E1x pinout Microsoft Excel® file attached to the IO_Definition document for the detailed mapping of the alternate function I/O pins.

20.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (MODER, OTYPER, OSPEEDR, PUPDR) to configure up to 16 I/Os. The MODER register is used to select the I/O mode (input, output, AF, analog). The OTYPER and OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

20.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (IDR and ODR). ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (IDR), a read-only register.

See *Section 20.4.6: Input data register (IDR) (x = A to I)* and *Section 20.4.7: Output data register (ODR) (x = A to I)* for the register descriptions.

20.3.5 I/O data bitwise handling

The bit set reset register (BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (ODR). The bit set reset register has twice the size of ODR.

Each bit in ODR corresponds to two control bits in BSRR: BS(i) and BR(i). When written to 1, bit BS(i) sets the corresponding ODR(i) bit. When written to 1, bit BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in BSRR does not have any effect on the corresponding bit in ODR. If there is an attempt to both set and reset a bit in BSRR, the set action takes priority.

Using the BSRR register to change the values of individual bits in ODR is a “one-shot” effect that does not lock the ODR bits. The ODR bits can always be accessed directly. The BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

20.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the LCKR register. The frozen registers are MODER, OTYPER, OSPEEDR, PUPDR, AFRL and AFRH.

To write the LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each LCKR bit freezes the corresponding bit in the control registers (MODER, OTYPER, OSPEEDR, PUPDR, AFRL and AFRH.

The LOCK sequence (refer to *Section 20.4.9: Configuration lock register (LCKR) (x = A to I)*) can only be performed using a word (32-bit long) access to the LCKR register due to the fact that LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in *Section 20.4.9: Configuration lock register (LCKR) (x = A to I)*.

20.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the AFRL and AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the SR5E1x pinout Microsoft Excel® file attached to the IO_Definition document.

20.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode.

Refer to *Chapter 26: Extended interrupt and event controller (EXTI)*.

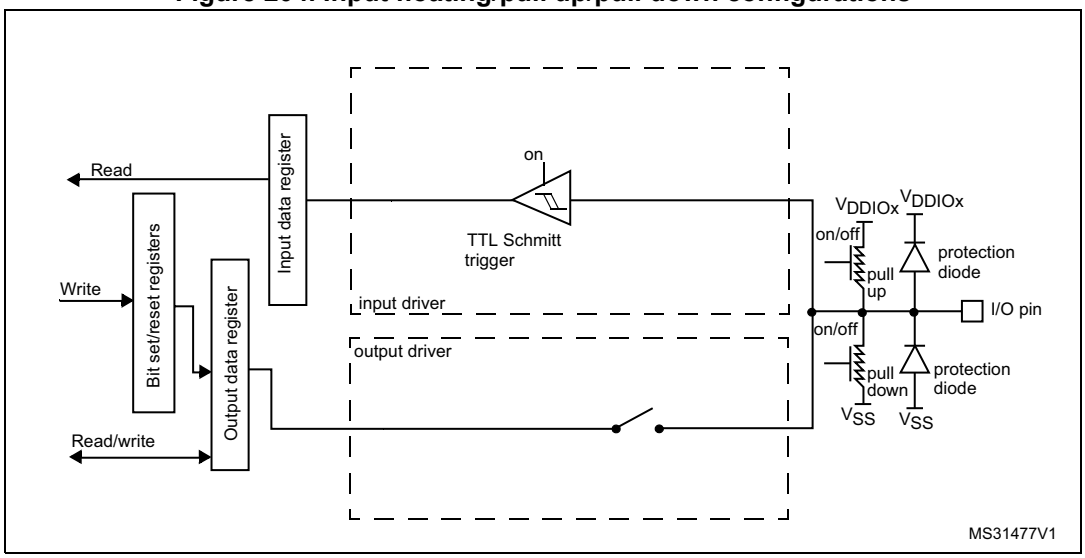
20.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

Figure 204 shows the input configuration of the I/O port bit.

Figure 204. Input floating/pull up/pull down configurations



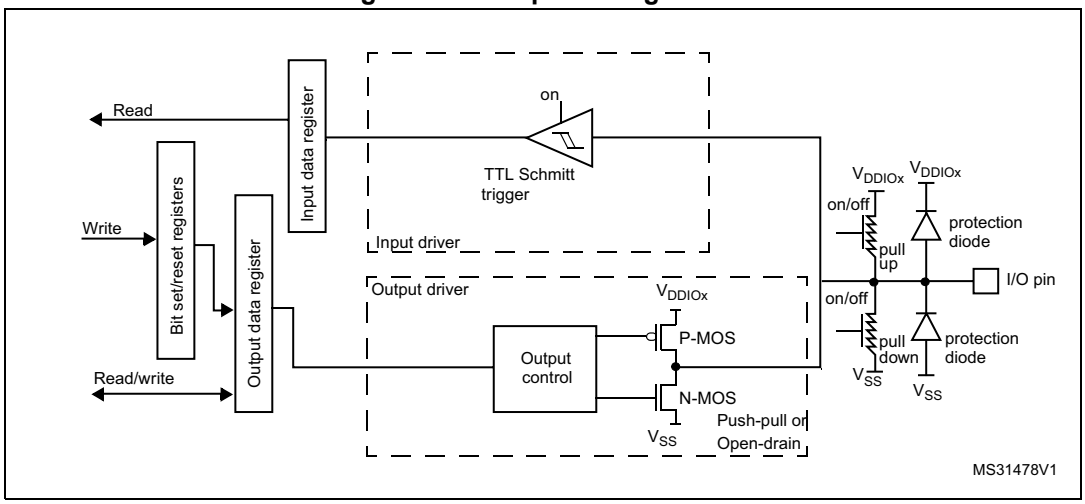
20.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

Figure 205 shows the output configuration of the I/O port bit.

Figure 205. Output configuration



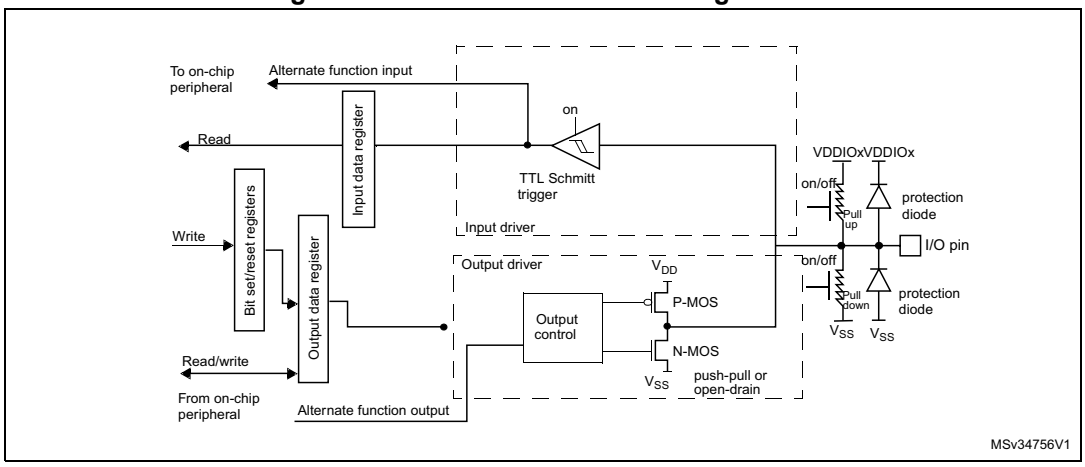
20.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

Figure 206 shows the Alternate function configuration of the I/O port bit.

Figure 206. Alternate function configuration



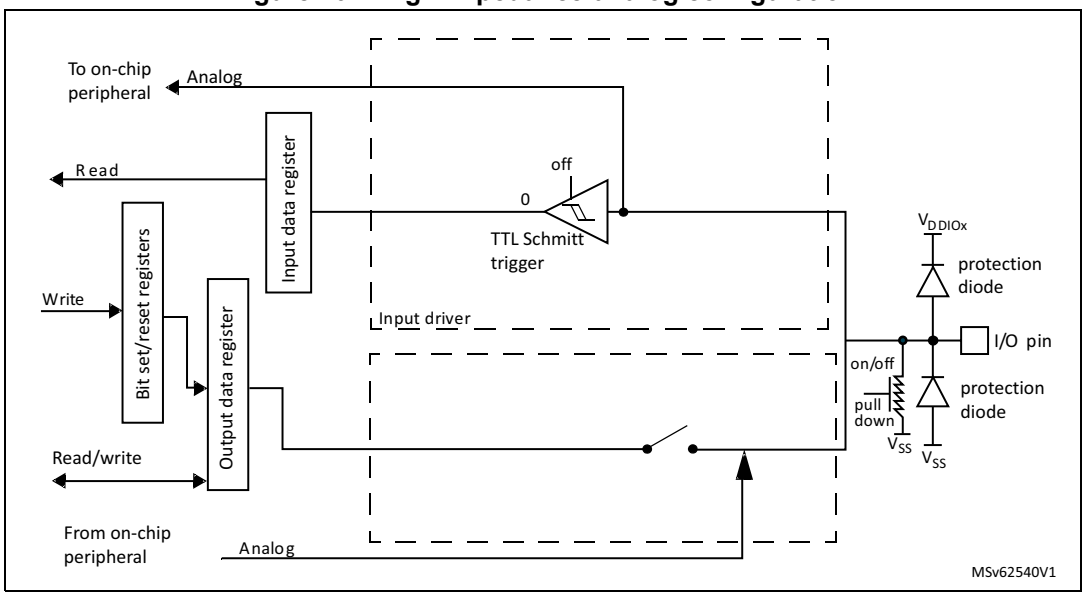
20.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up is disabled by Hardware. The weak pull-down is configurable.
- Read access to the input data register gets the value “0”

Figure 207 shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 207. High impedance-analog configuration



20.3.13 Safe Mode Configuration (SMC) of GPIOs

Each GPIO port has two registers to configure the Safe Mode of the I/Os:

- SAFESELR register enables the safe mode operation,
- SAFEVALR register selects the level (High/Low) that is driven on the pin when safe mode is activated.

The FCCU fault status is connected to the GPIO port SAFEMODE in order to activate the safe mode when the FCCU ERROUT[0] is asserted.

Example of SW procedure to configure SMC:

- FCCU configuration, for each fault that has to enable SMC:
  - configure the fault to generate an NMI (allows the NMI handler to manage proper actions)
  - enable the fault reporting on ERROUT
- GPIO configuration, for each GPIO pin:
  - enable/disable safe mode
  - select output level in safe mode

20.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to the table below.

The peripheral registers can be written in word, half word or byte mode.

20.4.1 GPIO memory map

Table 278. GPIO register memory map

Offset	Register name
0x00	*Mode register (MODER) (x =A to I)*
0x04	*Output type register (OTYPER) (x = A to I)*
0x08	*Output speed register (OSPEEDR) (x = A to I)*
0x0C	*Pull-up/pull-down register (PUPDR) (x = A to I)*
0x10	*Input data register (IDR) (x = A to I)*
0x14	*Output data register (ODR) (x = A to I)*
0x18	*Bit set/reset register (BSRR) (x = A to I)*
0x1C	*Configuration lock register (LCKR) (x = A to I)*
0x20	*Alternate function low register (AFRL) (x = A to I)*
0x24	*Alternate function high register (AFRH) (x = A to I)*
0x28	*Port Hysteresis Register (IHYSTR) (x = A to I)*
0x2C	*Port Input Buffer Enable Register (TRIGENR) (x = A to I)*
0x30	*Safe IO Selection Register (SAFESELR) (x = A to I)*
0x34	*Safe IO Output Value Register (SAFEVALR) (x = A to I)*

20.4.2 Mode register (MODER) (x =A to I)

Address offset: 0x00

Reset value: 0xFFFB FFFF (for port A)

Reset value: 0xFFFF FE80 (for port G)

Reset value: 0x000F FFFF (for ports I)

Reset value: 0xFFFF FFFF (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODE[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

20.4.3 Output type register (OTYPER) (x = A to I)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OT[15:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

20.4.4 Output speed register (OSPEEDR) (x = A to I)

Address offset: 0x08

Reset value: 0x0000 0300 (for port G)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEED[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output speed.
00: Low speed (slow configuration)
01: Medium speed (medium configuration)
10: High speed (fast configuration)
11: Very high speed (very fast configuration)

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

20.4.5 Pull-up/pull-down register (PUPDR) (x = A to I)

Address offset: 0x0C

Reset value: 0x0000 0040 (for port G)

Reset value: 0x0000 0000 (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPD[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

20.4.6 Input data register (IDR) (x = A to I)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ID[15:0]:** Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

20.4.7 Output data register (ODR) (x = A to I)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OD[15:0]:** Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the BSRR register (x = A..F).

20.4.8 Bit set/reset register (BSRR) (x = A to I)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit
1: Resets the corresponding ODx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BS[15:0]:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit
1: Sets the corresponding ODx bit

20.4.9 Configuration lock register (LCKR) (x = A to I)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the

LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = ‘1’ + LCKR[15:0]

WR LCKR[16] = ‘0’ + LCKR[15:0]

WR LCKR[16] = ‘1’ + LCKR[15:0]

RD LCKR

RD LCKR[16] = ‘1’ (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns ‘1’ until the next MCU reset or peripheral reset.

Bits 15:0 **LCK[15:0]:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is ‘0.

0: Port configuration not locked

1: Port configuration locked

20.4.10 Alternate function low register (AFRL) (x = A to I)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSEL[7:0][3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

20.4.11 Alternate function high register (AFRH) (x = A to I)

Address offset: 0x24

Reset value: 0x0000 0050 (for port A)

Reset value: 0x0000 0000 (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSEL[15:8][3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

20.4.12 Port Hysteresis Register (IHYSTR) (x = A to I)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IHYSTR15 [AUTO:TTL]		IHYSTR14 [AUTO:TTL]		IHYSTR13 [AUTO:TTL]		IHYSTR12 [AUTO:TTL]		IHYSTR11 [AUTO:TTL]		IHYSTR10 [AUTO:TTL]		IHYSTR9 [AUTO:TTL]		IHYSTR8 [AUTO:TTL]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IHYSTR7 [AUTO:TTL]		IHYSTR6 [AUTO:TTL]		IHYSTR5 [AUTO:TTL]		IHYSTR4 [AUTO:TTL]		IHYSTR3 [AUTO:TTL]		IHYSTR2 [AUTO:TTL]		IHYSTR1 [AUTO:TTL]		IHYSTR0 [AUTO:TTL]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IHYSTR[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are used to configure the I/O cells compatible levels of Schmitt input (CMOS/TTL/automotive).
00: CMOS
01: TTL
10: AUTO
11: TTL

20.4.13 Port Input Buffer Enable Register (TRIGENR) (x = A to I)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR33	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TR[15:0]:** Port configuration I/O pin y (y = 15 to 0)

These bits are used to enable the input buffer (TRIGEN[i-1:0]) of the I/O cells when I/O direction is configured as OUTPUT (MODER(i)[1:0] = “01”) or ALTERNATE FUNCTION (MODER(i)[1:0] = “10”) in MODER register. When I/O direction is configured as INPUT (MODER(i)[1:0] = “00”) the input buffer is always enabled, regardless of TRIGENR register content; when I/O direction is configured as ANALOG INOUT (MODER(i)[1:0] = “11”) the input buffer is always disabled, regardless of TRIGENR register content.

0: Input buffer disabled
1: Input buffer enabled

20.4.14 Safe IO Selection Register (SAFESELR) (x = A to I)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SFS15	SFS14	SFS13	SFS12	SFS11	SFS10	SFS9	SFS8	SFS7	SFS6	SFS5	SFS4	SFS3	SFS2	SFS1	SFS0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SFS[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are used to select the I/Os cells to be configured in “safe state” mode when the SAFESTATE input is driven to high logic level. In “safe state” mode the selected IO is configured as output push-pull and the input buffer is disabled. The output value is the one set in SAFEVAL register.

20.4.15 Safe IO Output Value Register (SAFEVALR) (x = A to I)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SFV15	SFV14	SFV13	SFV12	SFV11	SFV10	SFV9	SFV8	SFV7	SVS6	SFV5	SFV4	SFV3	SFV2	SFV1	SFV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SFV[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are used to select the I/Os cells to be configured in “safe state” mode when the SAFESTATE input is driven to high logic level. In “safe state” mode the selected IO is configured as output push-pull and the input buffer is disabled. The output value is the one set in SAFEVAL register.

21 System configuration controller (SYSCFG)

21.1 SYSCFG main features

SYSCFG controller contains the factory programmed identification registers for SR5E1x. Moreover, it provides configuration registers to set up specific functionalities of the device. The main functionalities controlled by the SYSCFG are the following:

- Managing the external interrupt line connection to the GPIOs
- Configuring the source for SDADCx external gating for DMA and interrupts
- Enabling BIAS analog block for SAR and SD ADCs
- Enabling and controlling filtering for FCCU Error Inputs pins
- Configuring the self-test voltage for comparators and SAR_ADCs
- Providing a SW trigger for HRTIMs
- Configuring the sources for the System Fault Inputs of HRTIMs

Glitch filters are present on EIN0 and EIN1 inputs and filter out pulses of width up to 50/75/100 µs. The duration configuration is done in the FCCU_FILTER1 and FCCU_FILTER2 registers of the SYSCFG module with the bit field FILTER_WIDTH. The filter becomes active as soon as the bit FCCU_FILTERx[FILTER_BYPASS] is de-asserted (default value).

The implementation is as follows:

1. Fault line assertion - If a Fault is asserted, then the output of the filter is asserted after the delay, equal to the programmed glitch width time. If the fault is de-asserted before the programmed glitch width time, then the output of the filter is not asserted and stays at the deactivated level (after having rejected the spurious glitch).
2. Fault line de-assertion - If a Fault, which was kept asserted for a time greater than the programmed pulse width is de-asserted (and glitch filter output is at asserted level now), then the output of the filter is de-asserted only after the delay equal to the programmed glitch width time. If the fault is again asserted before the programmed glitch width time, then the output of the filter is not de-asserted and stays at the active level (after having rejected the spurious glitch).

Note: The output of the filter becomes the effective fault input fed to the FCCU error management engine. Since there is an inherent latency involved, SW must either wait for the programmed delay value before trying to clear the fault status flag, FCCU failure input #45 or first set SYSCFG.FCCU_FILTERx[FILTER_BYPASS] to one and then try to clear the fault status flag. This bit must not be made '1' again, till the filter width time elapses, else the fault status is latched again.

21.2 SYSCFG register summary

Table 279. SYSCFG register list

Offset	Register name	Register title	Reset
0x0000	*MIDR1*	MCU ID Register 1	0x0000
0x0004	*MIDR2*	MCU ID Register 2	0x0000
0x0008	*EXTICR_3_0*	EXTI_3_0 Configuration register	0x0000

Table 279. SYSCFG register list (continued)

Offset	Register name	Register title	Reset
0x000C	*EXTICR_7_4*	EXTI_7_4 Configuration register	0x0000
0x0010	*EXTICR_11_8*	EXTI_11_8 Configuration register	0x0000
0x0014	*EXTICR_15_12*	EXTI_15_11 Configuration register	0x0000
0x0020	*SDADCx_EXT_GATE_SEL*	SDADCx EXT GATE Selection Register	0x0000
0x0030	*ADC_BIASEN*	ADC Bias Enable Configuration Register	0x0000
0x0040	*FCCU_FILTER1*	FCCU Glitch Filter 1 Configuration Register	0x0000
0x0044	*FCCU_FILTER2*	FCCU Glitch Filter 2 Configuration Register	0x0000
0x0200	*ST_VREF_SEL1*	COMP_BIAS Self-Test Configuration Register	0x0000
0x0204	*ST_VREF_SEL2*	SAR_BIAS Self-Test Configuration Register	0x0000
0x0300	*HRTIMx_UPDATE_EN*	HRTimerx Update Enable Register	0x0000
0x0304	*HRTIMxFLT_TIMxBRK_EN*	HRTimerx Fault and Timer Break Enable Register	0x0000

21.3 SYSCFG register description

MIDR1 MCU ID Register 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEVICE_INFO																RESERVED1	PACKAGE					RESERVED0		MAJOR_MASK				MINOR_MASK			
R																R	R					R		R				R			

Address: SYSCFGBaseAddress + 0x0000

Type: R

Reset: 0x0000

Description: This register contains identification information about the device.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:16] **DEVICE_INFO**: Stellar family, SR5E1x device part number

[15] **RESERVED1**: This field is reserved. It must be kept to zero by the software.

[14:10] **PACKAGE**: QFP176: 10001; QFP100: 01001; QFP144: 01101

[9:8] **RESERVED0**: This field is reserved. It must be kept to zero by the software

[7:4] **MAJOR_MASK**: Major mask revision

[3:0] **MINOR_MASK**: Minor mask revision

MIDR2 MCU ID Register 2

31	30 29 28 27	26 25 24 23	22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
SF	NVM_SIZE_1	NVM_SIZE_2	RESERVED1	SERIE	RESERVED0
R	R	R	R	R	R

Address: SYSCFGBaseAddress + 0x0004

Type: R

Reset: 0x0000

Description: This register contains identification information about the device.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31] **SF**: Manufacturer
0x0: Reserved
0x1: STMicroelectronics

[30:27] **NVM_SIZE_1**: coarse granularity for flash memory size. Needs to be combined with NVM_SIZE_2 to calculate the actual memory size
0x0: 16 KB
0x1: 32 KB
0x2: 64 KB
0x3: 128 KB
0x4: 256 KB
0x5: 512 KB
0x6: 1 MB
0x7: 2 MB
0x8: 4 MB
0x9: 8 MB
0xA: 16 MB
0xB: 32 MB
0xC: 64 MB
0xD: 128 MB
0xE: 256 MB
0xF: 512 MB

[26:23] **NVM_SIZE_2**: fine granularity for flash memory size. Needs to be combined with NVM_SIZE_1 to calculate the actual memory size
0x0: 0 KB
0x1: (FLASH_SIZE_1) / 8
0x2: 2 * (FLASH_SIZE_1) / 8
0x3: 3 * (FLASH_SIZE_1) / 8
0x4: 4 * (FLASH_SIZE_1) / 8
0x5: 5 * (FLASH_SIZE_1) / 8
0x6: 6 * (FLASH_SIZE_1) / 8
0x7: 7 * (FLASH_SIZE_1) / 8
0x8: 8 * (FLASH_SIZE_1) / 8
0x9: 9 * (FLASH_SIZE_1) / 8
0xA: 10 * (FLASH_SIZE_1) / 8
0xB: 11 * (FLASH_SIZE_1) / 8
0xC: 12 * (FLASH_SIZE_1) / 8
0xD: 13 * (FLASH_SIZE_1) / 8
0xE: 14 * (FLASH_SIZE_1) / 8
0xF: 15 * (FLASH_SIZE_1) / 8

[22:16] **RESERVED1**: This field is reserved. It must be kept to zero by the software

[15:8] **SERIE**: Serie E - ASCII character

[7:0] **RESERVED0**: This field is reserved. It must be kept to zero by the software

EXTICR_3_0 EXTI_3_0 Configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																EXTI3				EXTI2				EXTI1				EXTI0			
R																RW				RW				RW				RW			

Address: SYSCFGBaseAddress + 0x0008

Type: RW

Reset: 0x0000

Description: This register can be used to configure the exti selection (from 0 to 3).
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:16] **RESERVED**: These bits are reserved and they have to be kept to the default value

[15:12] **EXTI3**: EXTI3 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI3 mux out is PA3 (default value)
0x1: Line 1 selection: EXTI3 mux out is PB3
0x2: Line 2 selection: EXTI3 mux out is PC3
0x3: Line 3 selection: EXTI3 mux out is PD3
0x4: Line 4 selection: EXTI3 mux out is PE3
0x5: Line 5 selection: EXTI3 mux out is PF3
0x6: Line 6 selection: EXTI3 mux out is PG3
0x7: Line 7 selection: EXTI3 mux out is PH3
0x8: Line 8 selection: EXTI3 mux out is PI3
0x9: Line 9 selection: EXTI3 mux out is tied to 'b0
0xA: Line 10 selection: EXTI3 mux out is tied to 'b0
0xB: Line 11 selection: EXTI3 mux out is tied to 'b0
0xC: Line 12 selection: EXTI3 mux out is tied to 'b0
0xD: Line 13 selection: EXTI3 mux out is tied to 'b0
0xE: Line 14 selection: EXTI3 mux out is tied to 'b0
0xF: Line 15 selection: EXTI3 mux out is tied to 'b0

[11:8] **EXTI2**: EXTI2 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI2 mux out is PA2 (default value)
0x1: Line 1 selection: EXTI2 mux out is PB2
0x2: Line 2 selection: EXTI2 mux out is PC2
0x3: Line 3 selection: EXTI2 mux out is PD2
0x4: Line 4 selection: EXTI2 mux out is PE2
0x5: Line 5 selection: EXTI2 mux out is PF2
0x6: Line 6 selection: EXTI2 mux out is PG2
0x7: Line 7 selection: EXTI2 mux out is PH2
0x8: Line 8 selection: EXTI2 mux out is PI2
0x9: Line 9 selection: EXTI2 mux out is tied to 'b0
0xA: Line 10 selection: EXTI2 mux out is tied to 'b0
0xB: Line 11 selection: EXTI2 mux out is tied to 'b0
0xC: Line 12 selection: EXTI2 mux out is tied to 'b0
0xD: Line 13 selection: EXTI2 mux out is tied to 'b0
0xE: Line 14 selection: EXTI2 mux out is tied to 'b0
0xF: Line 15 selection: EXTI2 mux out is tied to 'b0

[7:4] **EXTI1**: EXTI1 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI1 mux out is PA1 (default value)
0x1: Line 1 selection: EXTI1 mux out is PB1
0x2: Line 2 selection: EXTI1 mux out is PC1
0x3: Line 3 selection: EXTI1 mux out is PD1
0x4: Line 4 selection: EXTI1 mux out is PE1
0x5: Line 5 selection: EXTI1 mux out is PF1
0x6: Line 6 selection: EXTI1 mux out is PG1
0x7: Line 7 selection: EXTI1 mux out is PH1
0x8: Line 8 selection: EXTI1 mux out is PI1
0x9: Line 9 selection: EXTI1 mux out is tied to 'b0
0xA: Line 10 selection: EXTI1 mux out is tied to 'b0
0xB: Line 11 selection: EXTI1 mux out is tied to 'b0
0xC: Line 12 selection: EXTI1 mux out is tied to 'b0
0xD: Line 13 selection: EXTI1 mux out is tied to 'b0
0xE: Line 14 selection: EXTI1 mux out is tied to 'b0
0xF: Line 15 selection: EXTI1 mux out is tied to 'b0

[3:0] **EXTI0**: EXTI0 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI0 mux out is PA0 (default value)
0x1: Line 1 selection: EXTI0 mux out is PB0
0x2: Line 2 selection: EXTI0 mux out is PC0
0x3: Line 3 selection: EXTI0 mux out is PD0
0x4: Line 4 selection: EXTI0 mux out is PE0
0x5: Line 5 selection: EXTI0 mux out is PF0
0x6: Line 6 selection: EXTI0 mux out is PG0
0x7: Line 7 selection: EXTI0 mux out is PH0
0x8: Line 8 selection: EXTI0 mux out is PI0
0x9: Line 9 selection: EXTI0 mux out is tied to 'b0
0xA: Line 10 selection: EXTI0 mux out is tied to 'b0
0xB: Line 11 selection: EXTI0 mux out is tied to 'b0
0xC: Line 12 selection: EXTI0 mux out is tied to 'b0
0xD: Line 13 selection: EXTI0 mux out is tied to 'b0
0xE: Line 14 selection: EXTI0 mux out is tied to 'b0
0xF: Line 15 selection: EXTI0 mux out is tied to 'b0

EXTICR_7_4 EXTI_7_4 Configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																EXTI7				EXTI6				EXTI5				EXTI4			
R																RW				RW				RW				RW			

Address: SYSCFGBaseAddress + 0x000C

Type: RW

Reset: 0x0000

Description: This register can be used to configure the exti selection (from 4 to 7).
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:16] **RESERVED**: These bits are reserved and they have to be kept to the default value

[15:12] **EXTI7**: EXTI7 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI7 mux out is PA7 (default value)
0x1: Line 1 selection: EXTI7 mux out is PB7
0x2: Line 2 selection: EXTI7 mux out is PC7
0x3: Line 3 selection: EXTI7 mux out is PD7
0x4: Line 4 selection: EXTI7 mux out is PE7
0x5: Line 5 selection: EXTI7 mux out is PF7
0x6: Line 6 selection: EXTI7 mux out is PG7
0x7: Line 7 selection: EXTI7 mux out is PH7
0x8: Line 8 selection: EXTI7 mux out is PI7
0x9: Line 9 selection: EXTI7 mux out is tied to 'b0
0xA: Line 10 selection: EXTI7 mux out is tied to 'b0
0xB: Line 11 selection: EXTI7 mux out is tied to 'b0
0xC: Line 12 selection: EXTI7 mux out is tied to 'b0
0xD: Line 13 selection: EXTI7 mux out is tied to 'b0
0xE: Line 14 selection: EXTI7 mux out is tied to 'b0
0xF: Line 15 selection: EXTI7 mux out is tied to 'b0

[11:8] **EXTI6**: EXTI6 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI6 mux out is PA6 (default value)
0x1: Line 1 selection: EXTI6 mux out is PB6
0x2: Line 2 selection: EXTI6 mux out is PC6
0x3: Line 3 selection: EXTI6 mux out is PD6
0x4: Line 4 selection: EXTI6 mux out is PE6
0x5: Line 5 selection: EXTI6 mux out is PF6
0x6: Line 6 selection: EXTI6 mux out is PG6
0x7: Line 7 selection: EXTI6 mux out is PH6
0x8: Line 8 selection: EXTI6 mux out is PI6
0x9: Line 9 selection: EXTI6 mux out is tied to 'b0
0xA: Line 10 selection: EXTI6 mux out is tied to 'b0
0xB: Line 11 selection: EXTI6 mux out is tied to 'b0
0xC: Line 12 selection: EXTI6 mux out is tied to 'b0
0xD: Line 13 selection: EXTI6 mux out is tied to 'b0
0xE: Line 14 selection: EXTI6 mux out is tied to 'b0
0xF: Line 15 selection: EXTI6 mux out is tied to 'b0

[7:4] **EXTI5**: EXTI5 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI5 mux out is PA5 (default value)
0x1: Line 1 selection: EXTI5 mux out is PB5
0x2: Line 2 selection: EXTI5 mux out is PC5
0x3: Line 3 selection: EXTI5 mux out is PD5
0x4: Line 4 selection: EXTI5 mux out is PE5
0x5: Line 5 selection: EXTI5 mux out is PF5
0x6: Line 6 selection: EXTI5 mux out is PG5
0x7: Line 7 selection: EXTI5 mux out is PH5
0x8: Line 8 selection: EXTI5 mux out is PI5
0x9: Line 9 selection: EXTI5 mux out is tied to 'b0
0xA: Line 10 selection: EXTI5 mux out is tied to 'b0
0xB: Line 11 selection: EXTI5 mux out is tied to 'b0
0xC: Line 12 selection: EXTI5 mux out is tied to 'b0
0xD: Line 13 selection: EXTI5 mux out is tied to 'b0
0xE: Line 14 selection: EXTI5 mux out is tied to 'b0
0xF: Line 15 selection: EXTI5 mux out is tied to 'b0

[3:0] **EXTI4**: EXTI4 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI4 mux out is PA4 (default value)
0x1: Line 1 selection: EXTI4 mux out is PB4
0x2: Line 2 selection: EXTI4 mux out is PC4
0x3: Line 3 selection: EXTI4 mux out is PD4
0x4: Line 4 selection: EXTI4 mux out is PE4
0x5: Line 5 selection: EXTI4 mux out is PF4
0x6: Line 6 selection: EXTI4 mux out is PG4
0x7: Line 7 selection: EXTI4 mux out is PH4
0x8: Line 8 selection: EXTI4 mux out is PI4
0x9: Line 9 selection: EXTI4 mux out is tied to 'b0
0xA: Line 10 selection: EXTI4 mux out is tied to 'b0
0xB: Line 11 selection: EXTI4 mux out is tied to 'b0
0xC: Line 12 selection: EXTI4 mux out is tied to 'b0
0xD: Line 13 selection: EXTI4 mux out is tied to 'b0
0xE: Line 14 selection: EXTI4 mux out is tied to 'b0
0xF: Line 15 selection: EXTI4 mux out is tied to 'b0

EXTICR_11_8 EXTI_11_8 Configuration register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
RESERVED	EXTI11	EXTI10	EXTI9	EXTI8
R	RW	RW	RW	RW

Address: SYSCFGBaseAddress + 0x0010

Type: RW

Reset: 0x0000

Description: This register can be used to configure the exti selection (from 8 to 11).
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:16] **RESERVED**: These bits are reserved and they have to be kept to the default value

[15:12] **EXTI11**: EXTI11 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI11 mux out is PA11 (default value)
0x1: Line 1 selection: EXTI11 mux out is PB11
0x2: Line 2 selection: EXTI11 mux out is PC11
0x3: Line 3 selection: EXTI11 mux out is PD11
0x4: Line 4 selection: EXTI11 mux out is PE11
0x5: Line 5 selection: EXTI11 mux out is PF11
0x6: Line 6 selection: EXTI11 mux out is PG11
0x7: Line 7 selection: EXTI11 mux out is PH11
0x8: Line 8 selection: EXTI11 mux out is PI11
0x9: Line 9 selection: EXTI11 mux out is tied to 'b0
0xA: Line 10 selection: EXTI11 mux out is tied to 'b0
0xB: Line 11 selection: EXTI11 mux out is tied to 'b0
0xC: Line 12 selection: EXTI11 mux out is tied to 'b0
0xD: Line 13 selection: EXTI11 mux out is tied to 'b0
0xE: Line 14 selection: EXTI11 mux out is tied to 'b0
0xF: Line 15 selection: EXTI11 mux out is tied to 'b0

[11:8] **EXTI10**: EXTI10 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI10 mux out is PA10 (default value)
0x1: Line 1 selection: EXTI10 mux out is PB10
0x2: Line 2 selection: EXTI10 mux out is PC10
0x3: Line 3 selection: EXTI10 mux out is PD10
0x4: Line 4 selection: EXTI10 mux out is PE10
0x5: Line 5 selection: EXTI10 mux out is PF10
0x6: Line 6 selection: EXTI10 mux out is PG10
0x7: Line 7 selection: EXTI10 mux out is PH10
0x8: Line 8 selection: EXTI10 mux out is PI10
0x9: Line 9 selection: EXTI10 mux out is tied to 'b0
0xA: Line 10 selection: EXTI10 mux out is tied to 'b0
0xB: Line 11 selection: EXTI10 mux out is tied to 'b0
0xC: Line 12 selection: EXTI10 mux out is tied to 'b0
0xD: Line 13 selection: EXTI10 mux out is tied to 'b0
0xE: Line 14 selection: EXTI10 mux out is tied to 'b0
0xF: Line 15 selection: EXTI10 mux out is tied to 'b0

[7:4] **EXTI9**: EXTI9 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI9 mux out is PA9 (default value)
0x1: Line 1 selection: EXTI9 mux out is PB9
0x2: Line 2 selection: EXTI9 mux out is PC9
0x3: Line 3 selection: EXTI9 mux out is PD9
0x4: Line 4 selection: EXTI9 mux out is PE9
0x5: Line 5 selection: EXTI9 mux out is PF9
0x6: Line 6 selection: EXTI9 mux out is PG9
0x7: Line 7 selection: EXTI9 mux out is PH9
0x8: Line 8 selection: EXTI9 mux out is PI9
0x9: Line 9 selection: EXTI9 mux out is tied to 'b0
0xA: Line 10 selection: EXTI9 mux out is tied to 'b0
0xB: Line 11 selection: EXTI9 mux out is tied to 'b0
0xC: Line 12 selection: EXTI9 mux out is tied to 'b0
0xD: Line 13 selection: EXTI9 mux out is tied to 'b0
0xE: Line 14 selection: EXTI9 mux out is tied to 'b0
0xF: Line 15 selection: EXTI9 mux out is tied to 'b0

[3:0] **EXTI8**: EXTI8 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI8 mux out is PA8 (default value)
0x1: Line 1 selection: EXTI8 mux out is PB8
0x2: Line 2 selection: EXTI8 mux out is PC8
0x3: Line 3 selection: EXTI8 mux out is PD8
0x4: Line 4 selection: EXTI8 mux out is PE8
0x5: Line 5 selection: EXTI8 mux out is PF8
0x6: Line 6 selection: EXTI8 mux out is PG8
0x7: Line 7 selection: EXTI8 mux out is PH8
0x8: Line 8 selection: EXTI8 mux out is PI8
0x9: Line 9 selection: EXTI8 mux out is tied to 'b0
0xA: Line 10 selection: EXTI8 mux out is tied to 'b0
0xB: Line 11 selection: EXTI8 mux out is tied to 'b0
0xC: Line 12 selection: EXTI8 mux out is tied to 'b0
0xD: Line 13 selection: EXTI8 mux out is tied to 'b0
0xE: Line 14 selection: EXTI8 mux out is tied to 'b0
0xF: Line 15 selection: EXTI8 mux out is tied to 'b0

EXTICR_15_12 EXTI_15_11 Configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																EXTI15				EXTI14				EXTI13				EXTI12			
R																RW				RW				RW				RW			

Address: SYSCFGBaseAddress + 0x0014

Type: RW

Reset: 0x0000

Description: This register can be used to configure the exti selection (from 12 to 15).
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:16] **RESERVED**: These bits are reserved and they have to be kept to the default value

[15:12] **EXTI15**: EXTI15 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI15 mux out is PA15 (default value)
0x1: Line 1 selection: EXTI15 mux out is PB15
0x2: Line 2 selection: EXTI15 mux out is PC15
0x3: Line 3 selection: EXTI15 mux out is PD15
0x4: Line 4 selection: EXTI15 mux out is PE15
0x5: Line 5 selection: EXTI15 mux out is PF15
0x6: Line 6 selection: EXTI15 mux out is PG15
0x7: Line 7 selection: EXTI15 mux out is PH15
0x8: Line 8 selection: EXTI15 mux out is PI15
0x9: Line 9 selection: EXTI15 mux out is tied to 'b0
0xA: Line 10 selection: EXTI15 mux out is tied to 'b0
0xB: Line 11 selection: EXTI15 mux out is tied to 'b0
0xC: Line 12 selection: EXTI15 mux out is tied to 'b0
0xD: Line 13 selection: EXTI15 mux out is tied to 'b0
0xE: Line 14 selection: EXTI15 mux out is tied to 'b0
0xF: Line 15 selection: EXTI15 mux out is tied to 'b0

[11:8] **EXTI14**: EXTI14 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI14 mux out is PA14 (default value)
0x1: Line 1 selection: EXTI14 mux out is PB14
0x2: Line 2 selection: EXTI14 mux out is PC14
0x3: Line 3 selection: EXTI14 mux out is PD14
0x4: Line 4 selection: EXTI14 mux out is PE14
0x5: Line 5 selection: EXTI14 mux out is PF14
0x6: Line 6 selection: EXTI14 mux out is PG14
0x7: Line 7 selection: EXTI14 mux out is PH14
0x8: Line 8 selection: EXTI14 mux out is PI14
0x9: Line 9 selection: EXTI14 mux out is tied to 'b0
0xA: Line 10 selection: EXTI14 mux out is tied to 'b0
0xB: Line 11 selection: EXTI14 mux out is tied to 'b0
0xC: Line 12 selection: EXTI14 mux out is tied to 'b0
0xD: Line 13 selection: EXTI14 mux out is tied to 'b0
0xE: Line 14 selection: EXTI14 mux out is tied to 'b0
0xF: Line 15 selection: EXTI14 mux out is tied to 'b0

[7:4] **EXTI13**: EXTI13 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI13 mux out is PA13 (default value)
0x1: Line 1 selection: EXTI13 mux out is PB13
0x2: Line 2 selection: EXTI13 mux out is PC13
0x3: Line 3 selection: EXTI13 mux out is PD13
0x4: Line 4 selection: EXTI13 mux out is PE13
0x5: Line 5 selection: EXTI13 mux out is PF13
0x6: Line 6 selection: EXTI13 mux out is PG13
0x7: Line 7 selection: EXTI13 mux out is PH13
0x8: Line 8 selection: EXTI13 mux out is PI13
0x9: Line 9 selection: EXTI13 mux out is tied to 'b0
0xA: Line 10 selection: EXTI13 mux out is tied to 'b0
0xB: Line 11 selection: EXTI13 mux out is tied to 'b0
0xC: Line 12 selection: EXTI13 mux out is tied to 'b0
0xD: Line 13 selection: EXTI13 mux out is tied to 'b0
0xE: Line 14 selection: EXTI13 mux out is tied to 'b0
0xF: Line 15 selection: EXTI13 mux out is tied to 'b0

[3:0] **EXTI12**: EXTI12 selection field. These bits must be written and cleared by the software
0x0: Line 0 selection: EXTI12 mux out is PA12 (default value)
0x1: Line 1 selection: EXTI12 mux out is PB12
0x2: Line 2 selection: EXTI12 mux out is PC12
0x3: Line 3 selection: EXTI12 mux out is PD12
0x4: Line 4 selection: EXTI12 mux out is PE12
0x5: Line 5 selection: EXTI12 mux out is PF12
0x6: Line 6 selection: EXTI12 mux out is PG12
0x7: Line 7 selection: EXTI12 mux out is PH12
0x8: Line 8 selection: EXTI12 mux out is PI12
0x9: Line 9 selection: EXTI12 mux out is tied to 'b0
0xA: Line 10 selection: EXTI12 mux out is tied to 'b0
0xB: Line 11 selection: EXTI12 mux out is tied to 'b0
0xC: Line 12 selection: EXTI12 mux out is tied to 'b0
0xD: Line 13 selection: EXTI12 mux out is tied to 'b0
0xE: Line 14 selection: EXTI12 mux out is tied to 'b0
0xF: Line 15 selection: EXTI12 mux out is tied to 'b0

SDADCx_EXT_GATE_SEL SDADCx EXT GATE Selection Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED1												SDADC2_EXT_GATE_SEL				RESERVED0												SDADC1_EXT_GATE_SEL			
R												RW				R												RW			

Address: SYSCFGBaseAddress + 0x0020

Type: RW

Reset: 0x0000

Description: This register can be used to configure the source for SDADCx external gating for DMA and interrupts - see SDADC integration guide for further details.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:20] **RESERVED1**: These bits are reserved and they have to be kept to the default value

[19:16] **SDADC2_EXT_GATE_SEL**: EXT Gating Selection for SDADC2. These bits must be written and cleared by the software
0x0: Line 0 selection: TIM1_OC1 is selected
0x1: Line 1 selection: TIM1_OC5 is selected
0x2: Line 2 selection: TIM8_OC1 is selected
0x3: Line 3 selection: TIM8_OC5 is selected
0x4: Line 4 selection: TIM2_OC4 is selected
0x5: Line 5 selection: TIM3_OC4 is selected
0x6: Line 6 selection: TIM4_OC4 is selected
0x7: Line 7 selection: TIM5_OC4 is selected
0x8: Line 8 selection: TIM15_OC2 is selected
0x9: Line 9 selection: TE of HRTIMER1 is selected
0xA: Line 10 selection: TF of HRTIMER1 is selected
0xB: Line 11 selection: TE of HRTIMER2 is selected
0xC: Line 12 selection: TF of HRTIMER2 is selected
0xD: Line 13 selection: mux out is tied to 'b0
0xE: Line 14 selection: mux out is tied to 'b0
0xF: Line 15 selection: mux out is tied to 'b0

[15:4] **RESERVED0**: These bits are reserved and they have to be kept to the default value

[3:0] **SDADC1_EXT_GATE_SEL**: EXT Gating Selection for SDADC1. These bits must be written and cleared by the software

0x0: Line 0 selection: TIM1_OC1 is selected
0x1: Line 1 selection: TIM1_OC5 is selected
0x2: Line 2 selection: TIM8_OC1 is selected
0x3: Line 3 selection: TIM8_OC5 is selected
0x4: Line 4 selection: TIM2_OC4 is selected
0x5: Line 5 selection: TIM3_OC4 is selected
0x6: Line 6 selection: TIM4_OC4 is selected
0x7: Line 7 selection: TIM5_OC4 is selected
0x8: Line 8 selection: TIM15_OC2 is selected
0x9: Line 9 selection: TE of HRTIMER1 is selected
0xA: Line 10 selection: TF of HRTIMER1 is selected
0xB: Line 11 selection: TE of HRTIMER2 is selected
0xC: Line 12 selection: TF of HRTIMER2 is selected
0xD: Line 13 selection: mux out is tied to 'b0
0xE: Line 14 selection: mux out is tied to 'b0
0xF: Line 15 selection: mux out is tied to 'b0

ADC_BIASEN ADC Bias Enable Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED1																															GLOBAL_BIAS_EN
R																															RW

Address: SYSCFGBaseAddress + 0x0030

Type: RW

Reset: 0x0000

Description: This register can be used to enable the bias used for SAR and SD ADC.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:1] **RESERVED1**: These bits are reserved and they have to be kept to the default value

[0] **GLOBAL_BIAS_EN**: Global Bias Enable. This bit must be written and cleared by the software. Once it is enabled, the read operation provides a 0x1 after 2us which is the BIAS startup time. The software must check this bit; when it is equal to 0x1, the SAR and the SD ADCs can be used (BIAS VBG = 0.9VBG)

0x0: ADC Bias is enabled
0x1: ADC Bias is disabled

FCCU_FILTER1 FCCU Glitch Filter 1 Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FILTER_BYPASS	FILTER_WIDTH	RESERVED1																													
RW	RW	R																													

Address: SYSCFGBaseAddress + 0x0040

Type: RW

Reset: 0x0000

Description: This register can be used to configure FCCU even glitch filter.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31] **FILTER_BYPASS**: FCCU Glitch Filter 1 Bypass. This bit must be written and cleared by the software

0x0: FCCU Glitch Filter 1 is not bypassed

0x1: FCCU Glitch Filter 1 is bypassed

[30:29] **FILTER_WIDTH**: FCCU Glitch Filter 1 Width. This bit must be written and cleared by the software. Make sure that the filter width does not cross the FOSU timeout value else unintended reset may occur

0x0: Filters glitches up to 50 us

0x1: Filters glitches up to 75 us

0x1: (B_0x2) Filters glitches up to 100 us for 16 MHz IRC clock

0x1: (B_0x3) Filters glitches up to 100 us for 16 MHz IRC clock

[28:0] **RESERVED1**: These bits are reserved and they have to be kept to the default value

FCCU_FILTER2 FCCU Glitch Filter 2 Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FILTER_BYPASS	FILTER_WIDTH	RESERVED1																													
RW	RW	R																													

Address: SYSCFGBaseAddress + 0x0044

Type: RW

Reset: 0x0000

Description: This register can be used to configure FCCU odd glitch filter.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31] **FILTER_BYPASS**: FCCU Glitch Filter 2 Bypass. This bit must be written and cleared by the software
0x0: FCCU Glitch Filter 2 is not bypassed
0x1: FCCU Glitch Filter 2 is bypassed

[30:29] **FILTER_WIDTH**: FCCU Glitch Filter 2 Width. This bit must be written and cleared by the software. Make sure that the filter width does not cross the FOSU timeout value else unintended reset may occur
0x0: Filters glitches up to 50 us
0x1: Filters glitches up to 75 us
0x1: (B_0x2) Filters glitches up to 100 us for 16 MHz IRC clock
0x1: (B_0x3) Filters glitches up to 100 us for 16 MHz IRC clock

[28:0] **RESERVED1**: These bits are reserved and they have to be kept to the default value

ST_VREF_SEL1 COMP_BIAS Self-Test Configuration Register

31	30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15	14 13 12 11 10 9 8 7 6 5 4 3 2	1 0
COMPBIAS_EN	RESERVED1	SELFTEST_EN	RESERVED0	COMP_TESTVREF_SEL
RW	R	RW	R	RW

Address: SYSCFGBaseAddress + 0x0200

Type: RW

Reset: 0x0000

Description: This register can be used to configure the self-test for the comparators bias.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31] **COMPBIAS_EN**: COMP Bias Enable bit
0x0: COMP Bias is disabled - it is in power down
0x1: COMP Bias is enabled - out of power down phase

[30:16] **RESERVED1**: This field is reserved. It must be kept to zero by the software

[15] **SELFTEST_EN**: Self-test enable bit. Enabling must occur after the voltage has been selected
0x0: ST is disabled. COMP_BIAS output is HiZ
0x1: ST is enabled. COMP_BIAS output is selected through COMP_TESTVREF_SEL field. Enabling must occur after the voltage has been selected.

[14:2] **RESERVED0**: This field is reserved. It must be kept to zero by the software

[1:0] **COMP_TESTVREF_SEL**: VREF selection for self-test
0x0: VREL is selected
0x1: 1/3 * (VREH-VREFL) is selected
0x2: 2/3 * (VREH-VREFL) is selected
0x3: VREH is selected

ST_VREF_SEL2 SAR_BIAS Self-Test Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SARBIAS_EN	RESERVED1															SELFTEST_EN	RESERVED0													SAR_TESTVREF_SEL	
RW	R															RW	R													RW	

Address: SYSCFGBaseAddress + 0x0204

Type: RW

Reset: 0x0000

Description: This register can be used to configure the self-test for the SAR ADC bias.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31] **SARBIAS_EN**: SAR ADC Bias Enable bit
0x0: SAR ADC Bias is disabled - it is in power down
0x1: SAR ADC Bias is enabled - out of power down phase

[30:16] **RESERVED1**: This field is reserved. It must be kept to zero by the software.

[15] **SELFTEST_EN**: Self-test enable bit. Enabling must occur after the voltage has been selected
0x0: ST is disabled. SAR ADC BIAS output is HiZ
0x1: ST is enabled. SAR ADC BIAS output is selected through SAR_TESTVREF_SEL field. Enabling must occur after the voltage has been selected.

[14:2] **RESERVED0**: This field is reserved. It must be kept to zero by the software

[1:0] **SAR_TESTVREF_SEL**: VREF selection for self-test
0x0: VREL is selected
0x1: 1/3 * (VREH-VREFL) is selected
0x2: 2/3 * (VREH-VREFL) is selected
0x3: VREH is selected

HRTIMx_UPDATE_EN HRTimerx Update Enable Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED0																															UPDATE_EN
R																															RW

Address: SYSCFGBaseAddress + 0x0300

Type: RW

Reset: 0x0000

Description: This register can be used to send the trigger for HRTIMERx Update Enable signals.
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:1] **RESERVED0**: This field is reserved. It must be kept to zero by the software.

[0] **UPDATE_EN**: Update Enable trigger. This bit must be written to 1 to send the trigger to both the HR Timers - pulse one clock cycle long. It is automatically cleared by the hardware.

HRTIMxFLT_TIMxBRK_EN HRTimerx Fault and Timer Break Enable Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED0																												TIMx_SAFEMODE_EN	TIMx_FLT_EN	HRTIMERx_SAFEMODE_EN	HRTIMERx_FLT_EN
R																												RW	RW	RW	RW

Address: SYSCFGBaseAddress + 0x0304

Type: RW

Reset: 0x0000

Description: This register can be used to enable FCCU fault and RCC safe mode entry request bits for HRTIMERx and TIMx
access_size: 32
min_write_access_size: 32
min_read_access_size: 32

[31:4] **RESERVED0**: This field is reserved. It must be kept to zero by the software

[3] **TIMx_SAFEMODE_EN**: TIMx Safe Mode Request Enable bit

0x0: Safe mode request is disabled for TIM1, TIM8, TIM15, TIM16

0x1: Safe mode request is enabled for TIM1, TIM8, TIM15, TIM16

Note: the safe mode configuration (SMC) of the GPIOs, if enabled on the TIM pins, takes priority over the TIM safe mode request and the TIM pin is driven in the safe mode at the configured SAFEVAL level.

[2] **TIMx_FLT_EN**: TIMx FCCU Fault Enable bit

0x0: FCCU Fault is disabled for TIM1, TIM8, TIM15, TIM16

0x1: FCCU Fault is enabled for TIM1, TIM8, TIM15, TIM16

[1] **HRTIMERx_SAFEMODE_EN**: Configure the pads entry to safe mode when System received safe mode request.

0x0: Safe mode request is disabled for HRTimerx

0x1: Safe mode request is enabled for HRTimerx

Note: the safe mode configuration (SMC) of the GPIOs, if enabled on the HRTIM pins, takes priority over the HRTIM safe mode request and the HRTIM pin is driven in the safe mode at the configured SAFEVAL level.

[0] **HRTIMERx_FLT_EN**: Configure the pads entry to safe mode when FCCU is in fault state

0x0: FCCU Fault is disabled for HRTimerx

0x1: FCCU Fault is enabled for HRTimerx

22 System memory protection unit (SMPU)

22.1 Overview

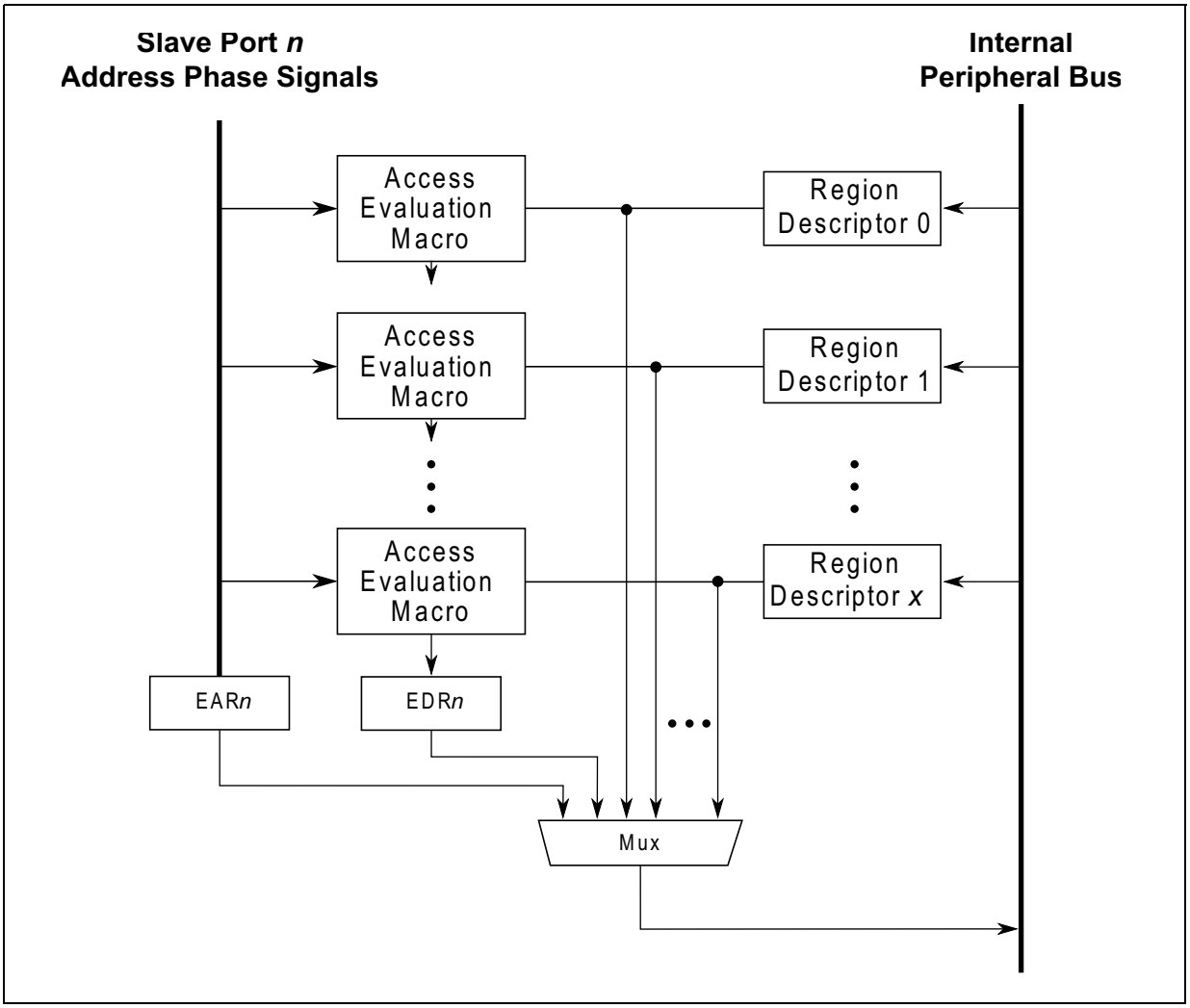
The system memory protection unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors and evaluates system bus transactions using preprogrammed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

22.2 Block diagram

A simplified block diagram of the SMPU module is shown in *Figure 208*. The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation macro shown as the replicated submodule block. The crossbar switch slave ports are shown on the left, the region descriptor registers are in the middle, and the peripheral bus interface is on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and detect protection violations.

For details of the access evaluation macro, refer to *Section 22.5.1: Access evaluation macro*.

Figure 208. SMPU block diagram



22.3 Features

The SMPU feature set includes:

- Supports up to 24 program-visible 128-bit region descriptors, accessible as four 32-bit words each.
  - Each region descriptor defines an arbitrarily sized space, aligned anywhere in memory.
    Region sizes can vary from a minimum of 1 byte to a maximum of (4 GB minus 1 byte).
  - Read/write access control permissions defined in region descriptor.
  - Cache-inhibit attribute indicator per region descriptor.
  - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues.
  - Priority given to granting permission over denying access for overlapping region descriptors.
- Supports as many as eight crossbar slave ports.
- Detects access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information.
- Global SMPU enables/disables control bit.

22.4 Memory map and register definition

22.4.1 Memory map

The programming model is partitioned into three groups: control registers, error capture registers, and the data structure containing the region descriptors.

The programming model can only be referenced using 32-bit accesses. Attempted references using different access sizes, to undefined (reserved) addresses, or with a non-supported access type (a write to a read-only register, or a read of a write-only register) generate an error termination.

Note: Refer to the device configuration details for any chip-specific register information for this module. For example, a specific instance of a module may support only up to 8 region descriptors.

Table 280. SMPU memory map

Address offset	Register name	Section
0x0000	CESR0	*Section 22.4.2.1*
0x0004	CESR1	*Section 22.4.2.2*
(n=0 to 15)		
0x8*n+0x0100	EARn	*Section 22.4.2.3*

Table 280. SMPU memory map (continued)

Address offset	Register name	Section
0x8*n+0x0104	EDRn	*Section 22.4.2.4*
(n=0 to 23)		
0x10*n+0x0400	RGDn_WORD0	*Section 22.4.2.5*
0x10*n+0x0404	RGDn_WORD1	*Section 22.4.2.6*
0x10*n+0x0408	RGDn_WORD2_FMT0	*Section 22.4.2.7*
0x10*n+0x040C	RGDn_WORD3	*Section 22.4.2.8*

22.4.2 Register descriptions

22.4.2.1 Control/Error Status Register 0 (CESR0)

Address: 0x0000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MERR															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	0	0	0	0	0	0	0	0	0	0	HRL			GVLD
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 209. Control/Error Status Register 0 (CESR0)

Table 281. CESR0 field descriptions

Field	Description
0:15 MERR	Master *n* error, where the bus master number matches the bit number Indicates a captured error in EAR*n* and EDR*n*. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing ‘1’ to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master *n*. 1 An error has occurred for bus master *n*.
16	Reserved This read-only bit is reserved and always has the value ‘1’.

Table 281. CESR0 field descriptions (continued)

Field	Description
28:30 HRL	Hardware revision level Specifies the SMPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled. **Note:** GVLD flag globally disables the SMPU even if any region has the RO flag of the RGDn_WORD3 register set to '1'.

22.4.2.2 Control/Error Status Register 1 (CESR1)

Address: 0x0004 Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEOVR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	0	0	0	0	0	0	0	0	0	0	NRGD			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 210. Control/Error Status Register 1 (CESR1)

Table 282. CESR1 field descriptions

Field	Description
0:15 MEOVR	Master *n* error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master *n* has occurred before the previous error was processed. The details of the first error are recorded in the EAR*n* and EDR*n* registers, and no information on subsequent errors is recorded until the associated CESR0.MERR flag is cleared. 0 No error overrun condition has been detected for bus master *n.* 1 An error overrun condition has been detected for bus master *n.*
16	Reserved This read-only bit is reserved and always has the value '1'.
28:31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

22.4.2.3 Error Address Register, Bus Master *n* (EAR*n*)

When the SMPU detects an access error on bus master *n*, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0.MERR is set. Additional information about the faulting access is captured in the corresponding EDR*n* at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1.MEOVR field until the original error is processed.

Address: 0x0100 + *n**0x8 (*n* = 0 to 15) Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 211. Error Address Register, Bus Master *n* (EAR*n*)

Table 283. EAR*n* field descriptions

Field	Description
0:31 EADDR	Error address Indicates the reference address from bus master *n* that generated the access error.

22.4.2.4 Error Detail Register, Bus Master *n* (EDR*n*)

When the SMPU detects an access error associated with bus master *n*, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0.MERR is set. Information on the faulting address is captured in the corresponding EAR*n* register at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1.MEOVR field until the original error is processed.

Address: 0x0104 + *n**0x8 (*n* = 0 to 15) Access: Supervisor read-only

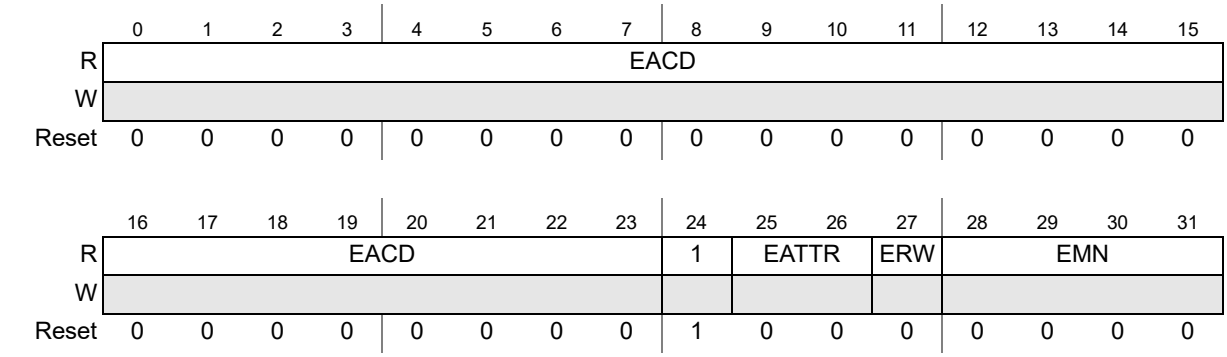


Figure 212. Error Detail Register, Bus Master *n* (EDR*n*)

Table 284. EDR*n* field descriptions

Field	Description
0:23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDR*n* contains a captured error and EACD is all zeroes, an access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24	Reserved This read-only bit is reserved and always has the value ‘1’.
25:26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference. 0 Read 1 Write
28:31 EMN	Error master number Indicates the logical bus master number of the faulting reference. This field is used to determine the bus master that generated the access error.

22.4.2.5 Region Descriptor *n*, Word 0 (RGD*n*_WORD0)

Address: 0x0400 + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 213. Region Descriptor *n*, Word 0 (RGD*n*_WORD0)

Table 285. RGD*n*_WORD0 field descriptions

Field	Description
0:31 SRTADDR	Start address Defines the byte start address of the memory region.

22.4.2.6 Region Descriptor *n*, Word 1 (RGD*n*_WORD1)

Address: 0x0404 + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 214. Region Descriptor *n*, Word 1 (RGD*n*_WORD1)

Table 286. RGD*n*_WORD1 field descriptions

Field	Description
0:31 ENDADDR	End address Defines the byte end address of the memory region. **Note:** The SMPU does not verify that ENDADDR ≥ SRTADDR.

22.4.2.7 Region Descriptor *n*, Word 2 Format 0 (RGD*n*_WORD2_FMT0)

RGD_WORD2 has two formats as determined by the RGD_WORD3.FMT field.

Note: RGD_WORD2_FMT0 applies when RGD_WORD3[FMT] = 0.

RGD_WORD2_FMT0 defines the access control rights of the memory region on a per master basis. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags per logical bus master:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch or an instruction fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) instruction.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor's valid bit.

Address: 0x0408 + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0P		M1P		M2P		M3P		M4P		M5P		M6P		M7P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8P		M9P		M10P		M11P		M12P		M13P		M14P		M15P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 215. Region Descriptor *n*, Word 2 Format 0 (RGD*n*_WORD2_FMT0)

Table 287. RGD*n*_WORD2_FMT0 field descriptions

Field	Description
0:1 M0P	Bus master 0 permissions 00 no access 01 R 10 W 11 R/W
2:3 M1P	Bus master 1 permission 00 no access 01 R 10 W 11 R/W
4:5 M2P	Bus master 2 permissions 00 no access 01 R 10 W 11 R/W

Table 287. RGD*n*_WORD2_FMT0 field descriptions (continued)

Field	Description
6:7 M3P	Bus master 3 permissions 00 no access 01 R 10 W 11 R/W
8:9 M4P	Bus master 4 permissions 00 no access 01 R 10 W 11 R/W
10:11 M5P	Bus master 5 permissions 00 no access 01 R 10 W 11 R/W
12:13 M6P	Bus master 6 permissions 00 no access 01 R 10 W 11 R/W
14:15 M7P	Bus master 7 permissions 00 no access 01 R 10 W 11 R/W
16:17 M8P	Bus master 8 permissions 00 no access 01 R 10 W 11 R/W
18:19 M9P	Bus master 9 permissions 00 no access 01 R 10 W 11 R/W
20:21 M10P	Bus master 10 permissions 00 no access 01 R 10 W 11 R/W

Table 287. RGD*n*_WORD2_FMT0 field descriptions (continued)

Field	Description
22:23 M11P	Bus master 11 permissions 00 no access 01 R 10 W 11 R/W
24:25 M12P	Bus master 12 permissions 00 no access 01 R 10 W 11 R/W
26:27 M13P	Bus master 13 permissions 00 no access 01 R 10 W 11 R/W
28:29 M14P	Bus master 14 permissions 00 no access 01 R 10 W 11 R/W
30:31 M15P	Bus master 15 permissions 00 no access 01 R 10 W 11 R/W

22.4.2.8 Region Descriptor *n*, Word 3 (RGD*n*_WORD3)

The final word of the SMPU region descriptor contains the valid bit, the format select (FMT), plus other configuration bits.

Address: 0x040C + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	FMT	RO	0	CI	VLD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 216. Region Descriptor n, Word 3 (RGDn_WORD3)

Table 288. RGD*n*_WORD3 field descriptions

Field	Description
27 FMT	Region descriptor format This bit selects the configuration format (FMT0 or FMT1) for this region descriptor. **Note:** A specific module instance of the SMPU may support only the FMT0 format. If so, the FMT field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies. 0 Use format 0 (RGD_WORD2_FMT0) 1 Use format 1 (RGD_WORD2_FMT1)
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. **Note:** Setting RO in an RGD locks all four words of the RGD until a system reset. RO flag does not affect the global valid disable flag (GLVD) of the CESR0 register. If the RO flag is set to '1', the software can still disable the SMPU by setting GLVD to '0'. 0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.
30 CI	Cache inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **Note:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size. 0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals that the region descriptor is valid. Any write to RGD*n*_WORD0–2 clears this bit. 0 Region descriptor is invalid. 1 Region descriptor is valid.

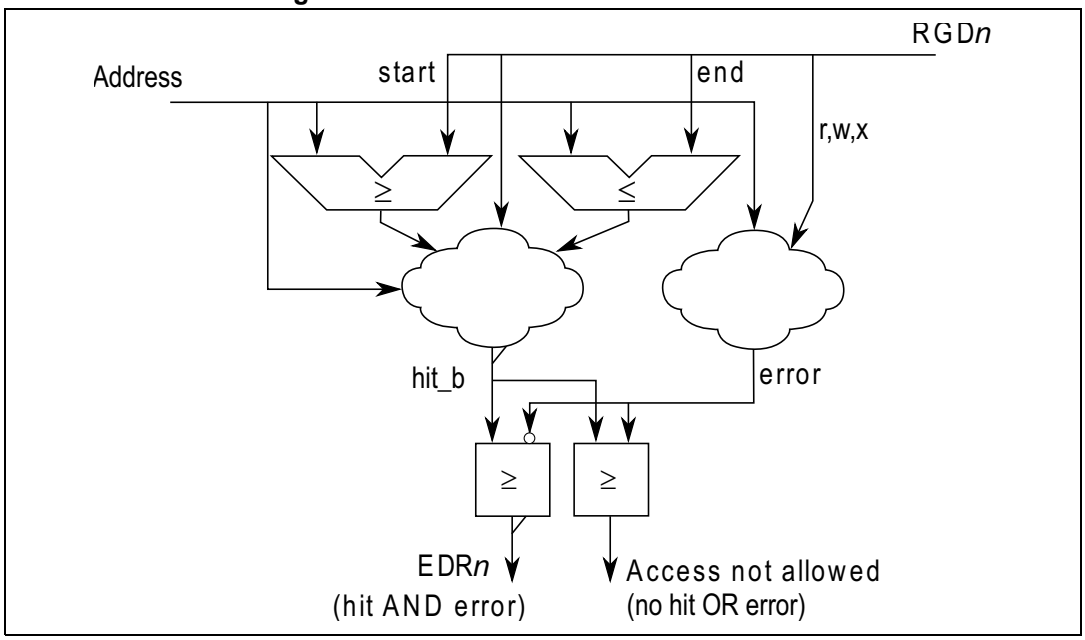
22.5 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

22.5.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in *Figure 217*, the access evaluation macro inputs the crossbar bus address phase signals and the contents of a region descriptor (RGD*n*), then performs two major functions: region hit determination and detection of an access protection violation. *Figure 217* shows a functional block diagram.

Figure 217. SMPU access evaluation macro



22.5.1.1 Hit determination

To determine if the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. The boolean equation (applied in parallel to the lower and higher addresses of a burst) for this portion of the hit determination is:

```
region_hit = ((addr[0:31] >= RGDn_Word0[SRTADDR]) &
      (addr_1[0:31] <= RGDn_Word1[ENDADDR])) &
RGDn_Word3[VLD]
```

Where addr is the current reference address, addr_1 is the last address in case of burst access (otherwise addr_1 = addr), RGDn_Word0.SRTADDR and RGDn_Word1.ENDADDR are the start and end addresses, and RGDn_Word3.VLD is the valid bit.

Note: The SMPU does not verify that ENDADDR ≥ SRTADDR.

22.5.1.2 Privilege violation determination

While the access evaluation macro is determining the region hit, the logic is also evaluating whether the current access is allowed by the permissions defined in the region descriptor. Using the master signal, a set of effective permissions is generated from the relevant fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions, using the specification shown in *Table 289*.

Table 289. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no read permission
	1	—	No, access is allowed

Table 289. Protection violation definition (continued)

Access type	Access permissions		Protection violation?
	r	w	
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

22.5.2 Putting it all together and error terminations

For each slave port monitored, the SMPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.
3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, refer to *Section 22.7*.

22.6 Initialization information

At system startup, load the required number of region descriptors, including setting RGD*n*_Word3.VLD. Setting CESR0.GVLD enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0.GVLD = 0). Next, load the appropriate region descriptors (RGD*n*), including the setting of the RGDn_Word3.VLD bits. Finally, set CESR0.GVLD to enable the entire module.

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

22.7 Application information

In an operational system, interfacing with the SMPU is generally classified into the following activities:

- Creating a new memory region: Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Modifying a region descriptor: Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word3 re-enables the region descriptor valid bit. The

hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

- Removing a region descriptor: Clearing RGD*n*_Word3.VLD deletes an existing region descriptor.
- Accessing the SMPU: Allocate a region descriptor to restrict SMPU access to supervisor mode from a specific master.
- Detecting an access error: The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0.MERR signals. The error registers contain the captured fault data.
- Overlapping region descriptors: Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

The following dual-core system example contains four bus masters: the two processors (CP0, CP1) and two DMA engines (DMA1, a traditional data movement engine transferring data between RAM and peripherals, and DMA2, a second engine transferring data to/from the RAM only). Consider the following region descriptor assignments:

Table 290. Overlapping region descriptor example

Region description	RGDn	CP0	CP1	DMA1	DMA2	System memory map space
CP0 code	0	rwx	r--	—	—	Flash
CP1 code	1	r--	rwx	—	—	
CP0 data & stack	2	rw-	—	—	—	RAM
CP0 → CP1 shared data	2	r--	r--	—	—	
CP1 → CP0 shared data	4					
CP1 data & stack	4	—	rw-	—	—	
Shared DMA data	5	rw-	rw-	rw	rw	
SMPU	6	rw-	rw-	—	—	Peripheral space
Peripherals	7	rw-	rw-	rw	—	

In this example, there are eight descriptors used to span nine regions in the three main spaces of the system memory map (flash, RAM, and peripheral space). Each region indicates the specific permissions for each of the four bus masters, and this definition provides an assigned set of shared, private, and executable memory spaces.

Of particular interest are the two overlapping spaces: region descriptors 2 & 3 and 3 & 4.

The space defined by RGD2 with no overlap is a private data and stack area that provides read/write access to CP0 only. The overlapping space between RGD2 and RGD3 defines a shared data space for passing data from CP0 to CP1, and the access controls are defined

by the logical OR of the two region descriptors. Thus, CP0 has (rw- | r--) = (rw-) permissions, while CP1 has (--- | r--) = (r--) permission in this space. Both DMA engines are excluded from this shared processor data region. The overlapping spaces between RGD3 and RGD4 defines another shared data space, this one for passing data from CP1 to CP0. For this overlapping space, CP0 has (r-- | ---) = (r--) permission, while CP1 has (rw- | r--) = (rw-) permission. The non-overlapped space of RGD4 defines a private data and stack area for CP1 only.

The space defined by RGD5 is a shared data region, accessible by all four bus masters. Finally, the slave peripheral space mapped onto the peripheral bus is partitioned into two regions: one containing the SMPU's programming model accessible only to the two processor cores and the remaining peripheral region accessible to both processors and the traditional DMA1 master.

This simple example is intended to show one possible application of the capabilities of the SMPU in a typical system.

23 Direct memory access controller (DMA)

23.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory and between memory and memory. Data can be quickly moved by DMA without any CPU action. This keeps CPU resources free for other operations.

The DMA controller combines a powerful dual AHB master bus architecture with independent FIFO to optimize the bandwidth of the system, based on a complex bus matrix architecture.

The two DMA controllers (DMA1, DMA2) have 8 input streams each, dedicated to managing memory access requests from one or more peripherals.

Each DMA stream is driven by one DMAMUX1 output channel (request). Any DMAMUX1 output request can be individually programmed in order to select the DMA request source signal, from any of the 151 available request input signals.

Refer to the *Section 24: DMA request multiplexer (DMAMUX)* for more information about the DMA requests and streams mapping.

Each DMA controller has an arbiter for handling the priority between DMA requests.

23.2 DMA main features

The main DMA features are:

- Dual AHB master bus architecture
- AHB slave programming interface supporting only 32-bit accesses
- 8 streams for each DMA controller, up to 151 channels (requests) per stream
- Four-word depth 32 first-in, first-out memory buffers (FIFOs) per stream, that can be used in FIFO mode or direct mode:
  - FIFO mode: with threshold level software selectable between 1/4, 1/2 or 3/4 of the FIFO size
  - Direct mode: each DMA request immediately initiates a transfer from/to the memory. When it is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads only one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.
- Each stream can be configured to be:
  - a regular channel that supports peripheral-to-memory, memory-to-peripheral and memory-to-memory transfers
  - a double buffer channel that also supports double buffering on the memory side
- Priorities between DMA stream requests are software-programmable (four levels consisting of very high, high, medium, low) or hardware in case of equality (for example, request 0 has priority over request 1)
- Each stream also supports software trigger for memory-to-memory transfers
- Each stream request can be selected among up to 151 possible channel requests. This selection is software-configurable by the DMAMUX1 and allows 107 peripherals to initiate DMA requests

- The number of data items to be transferred can be managed either by the DMA controller or by the peripheral:
  - DMA flow controller: the number of data items to be transferred is software-programmable from 1 to 65535
  - Peripheral flow controller: the number of data items to be transferred is unknown and controlled by the source or the destination peripheral that signals the end of the transfer by hardware

Note: SR5E1x has no peripheral able to use the peripheral flow controller mode.

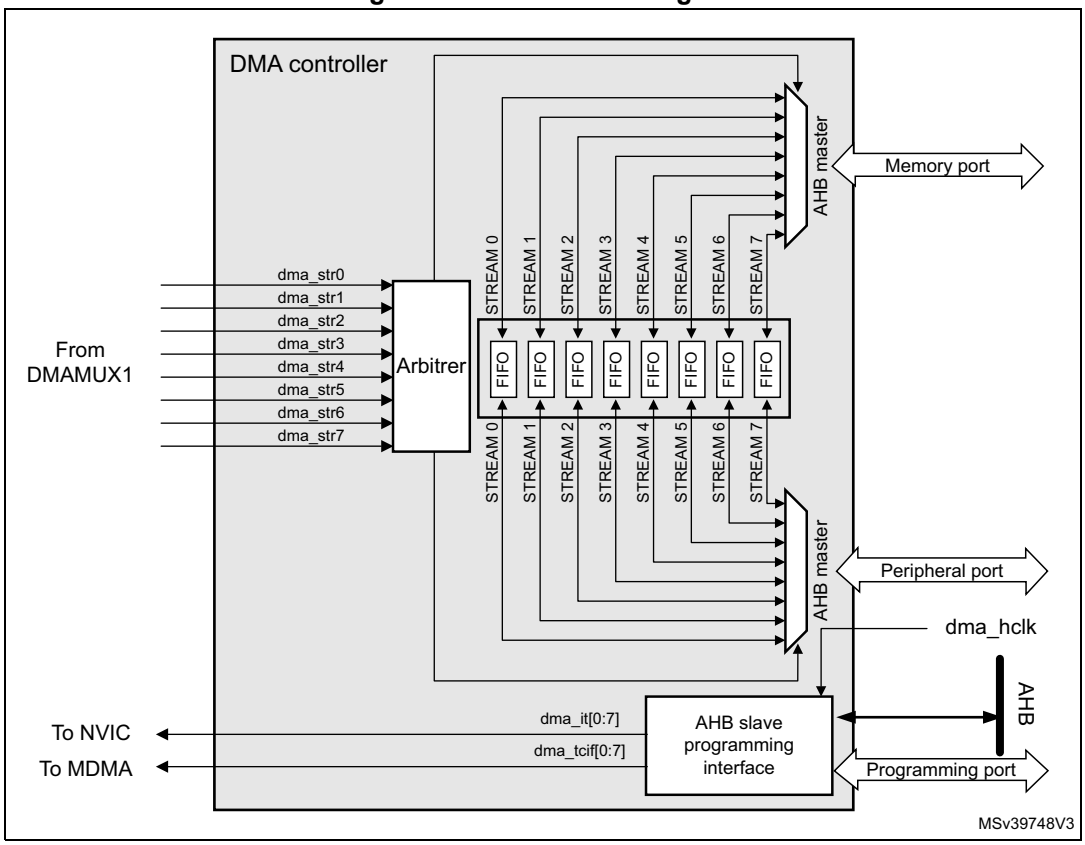
- Independent source and destination transfer width (byte, half-word, word): when the data widths of the source and destination are not equal, the DMA automatically packs/unpacks the necessary transfers to optimize the bandwidth. This feature is only available in FIFO mode
- Incrementing or non-incrementing addressing for source and destination
- Supports incremental burst transfers of 4, 8 or 16 beats. The size of the burst is software-configurable, usually equal to half the FIFO size of the peripheral
- Each stream supports circular buffer management
- 5 event flags (DMA half transfer, DMA transfer complete, DMA transfer error, DMA FIFO error, direct mode error) logically ORed together in a single interrupt request for each stream

23.3 DMA functional description

23.3.1 DMA block diagram

The figure below shows the block diagram of a DMA.

Figure 218. DMA block diagram



23.3.2 DMA internal signals

The table below shows the internal DMA signals.

Table 291. DMA internal input/output signals

Signal name	Signal type	Description
dma_hclk	Digital input	DMA AHB clock
dma_it[0:7]	Digital outputs	DMA stream [0:7] global interrupts
dma_str[0:7]	Digital input	DMA stream [0:7] requests

23.3.3 DMA overview

The DMA controller performs direct memory transfer: as an AHB master, the DMA controller can take the control of the AHB bus matrix to initiate AHB transactions.

The DMA controller carries out the following transactions:

- Peripheral-to-memory
- Memory-to-peripheral
- Memory-to-memory

The DMA controller provides two AHB master ports that are connected in an equivalent way to the bus matrix. From a functional point of view, they can be described as the AHB memory port, intended to be connected to memories and the AHB peripheral port, intended to be connected to peripherals. The AHB peripheral port has access to the memories to allow memory-to-memory transfers.

The AHB slave port is used to program the DMA controller (it supports only 32-bit accesses).

23.3.4 DMA transactions

A DMA transaction consists of a sequence of a given number of data transfers. The number of data items to be transferred and their width (8-bit, 16-bit or 32-bit) are software-programmable.

Each DMA transfer consists of three operations:

- A loading from the peripheral data register or a location in memory, addressed through the DMA_SxPAR or DMA_SxM0AR register
- A storage of the data loaded to the peripheral data register or a location in memory addressed through the DMA_SxPAR or DMA_SxM0AR register
- A post-decrement of the DMA_SxNDTR register, containing the number of transactions that still have to be performed

After an event, the peripheral sends a request signal to the DMA controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA controller accesses the peripheral, an Acknowledge signal is sent to the peripheral by the DMA controller. The peripheral releases its request as soon as it gets the Acknowledge signal from the DMA controller. Once the request has been deasserted by the peripheral, the DMA controller releases the Acknowledge signal. If there are more requests, the peripheral can initiate the next transaction.

23.3.5 DMA request mapping

The DMA request mapping from peripherals to DMA streams is described in *Section 24: DMA request multiplexer (DMAMUX)*.

23.3.6 Arbiter

An arbiter manages the 8 DMA stream requests based on their priority for each of the two AHB master ports (memory and peripheral ports) and launches the peripheral/memory access sequences.

Priorities are managed in two stages:

- Software: each stream priority can be configured in the DMA_SxCR register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: If two requests have the same software priority level, the stream with the lower number takes priority over the stream with the higher number. For example, stream 2 takes priority over stream 4.

23.3.7 DMA streams

Each of the eight DMA controller streams provides a unidirectional transfer link between a source and a destination.

Each stream can be configured to perform:

- Regular type transactions: memory-to-peripherals, peripherals-to-memory or memory-to-memory transfers
- Double-buffer type transactions: double buffer transfers using two memory pointers for the memory (while the DMA is reading/writing from/to a buffer, the application can write/read to/from the other buffer).

The amount of data to be transferred (up to 65535) is programmable and related to the source width of the peripheral that requests the DMA transfer connected to the peripheral AHB port. The register that contains the amount of data items to be transferred is decremented after each transaction.

23.3.8 Source, destination and transfer modes

Both source and destination transfers can address peripherals and memories in the entire 4-Gbyte area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The direction is configured using the DIR[1:0] bits in the DMA_SxCR register and offers three possibilities: memory-to-peripheral, peripheral-to-memory or memory-to-memory transfers.

The table below describes the corresponding source and destination addresses.

Table 292. Source and destination address

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00(1)	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR
01	Memory-to-peripheral	DMA_SxM0AR	DMA_SxPAR

Table 292. Source and destination address (continued)

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
10	Memory-to-memory	DMA_SxPAR	DMA_SxM0AR
11	Reserved	-	-

1. This configuration can also be used for peripheral to peripheral transfer as memory AHB port has access to peripheral address space through bus matrix (NIC) connections.

When the data width (programmed in the PSIZE or MSIZE bits in the DMA_SxCR register) is a half-word or a word, respectively, the peripheral or memory address written into the DMA_SxPAR or DMA_SxM0AR/M1AR registers has to be aligned on a word or half-word address boundary, respectively.

Peripheral-to-memory mode

Figure 219 describes this mode.

When this mode is enabled (by setting the bit EN in the DMA_SxCR register), each time a peripheral request occurs, the stream initiates a transfer from the source to fill the FIFO.

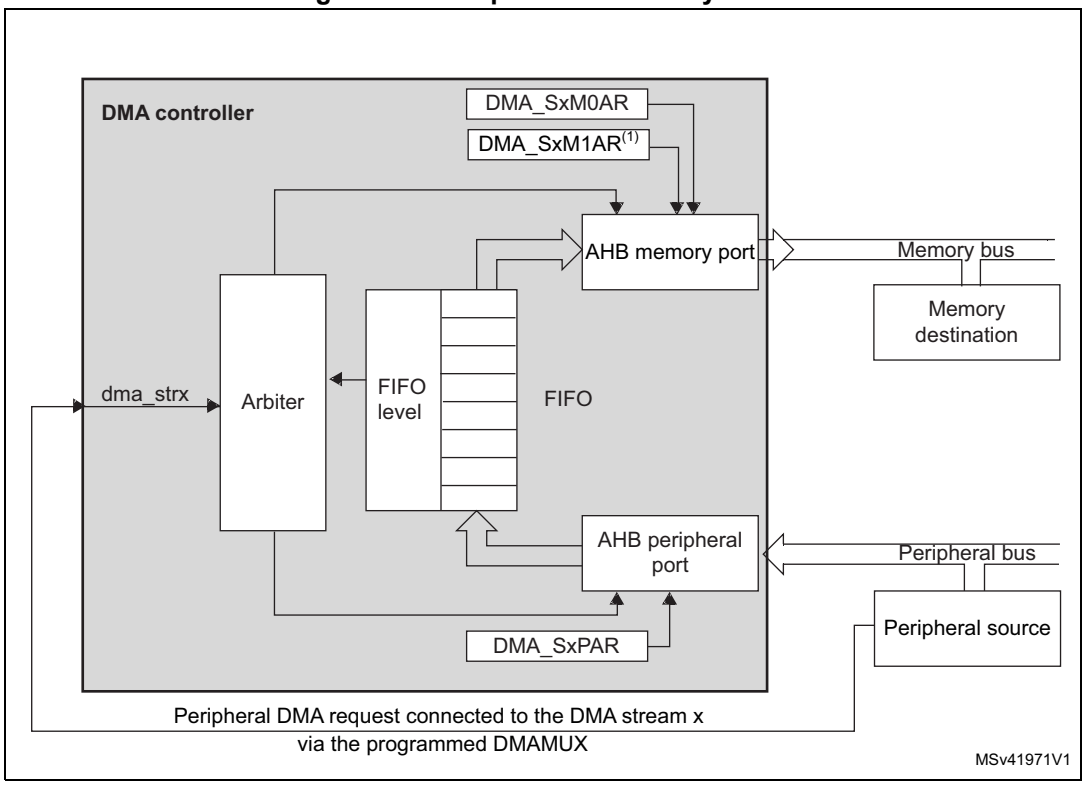
When the threshold level of the FIFO is reached, the contents of the FIFO are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is 0), the threshold level of the FIFO is not used: after each single data transfer from the peripheral to the FIFO, the corresponding data are immediately drained and stored into the destination.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 219. Peripheral-to-memory mode



1. For double-buffer mode.

Memory-to-peripheral mode

Figure 220 describes this mode.

When this mode is enabled (by setting the EN bit in the DMA_SxCR register), the stream immediately initiates transfers from the source to entirely fill the FIFO.

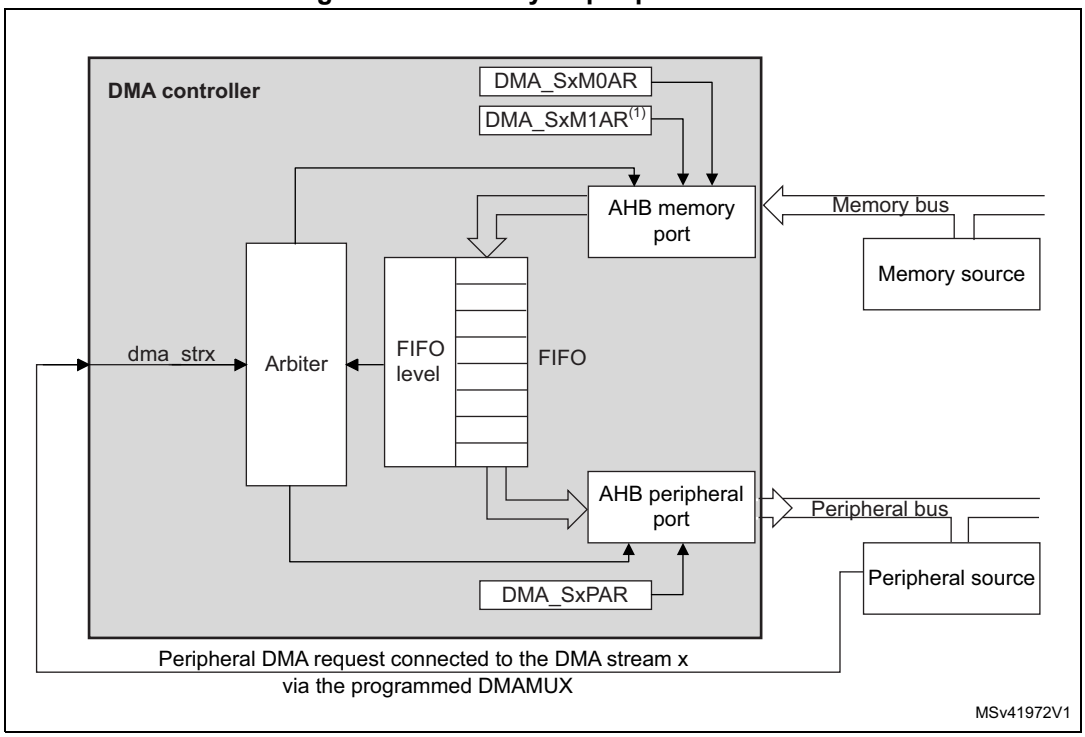
Each time a peripheral request occurs, the contents of the FIFO are drained and stored into the destination. When the level of the FIFO is lower than or equal to the predefined threshold level, the FIFO is fully reloaded with data from the memory.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is 0), the threshold level of the FIFO is not used. Once the stream is enabled, the DMA preloads the first data to transfer into an internal FIFO. As soon as the peripheral requests a data transfer, the DMA transfers the preloaded value into the configured destination. It then reloads again the empty internal FIFO with the next data to be transfer. The preloaded data size corresponds to the value of the PSIZE bit field in the DMA_SxCR register.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 220. Memory-to-peripheral mode



1. For double-buffer mode.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This is the memory-to-memory mode, described in *Figure 221*.

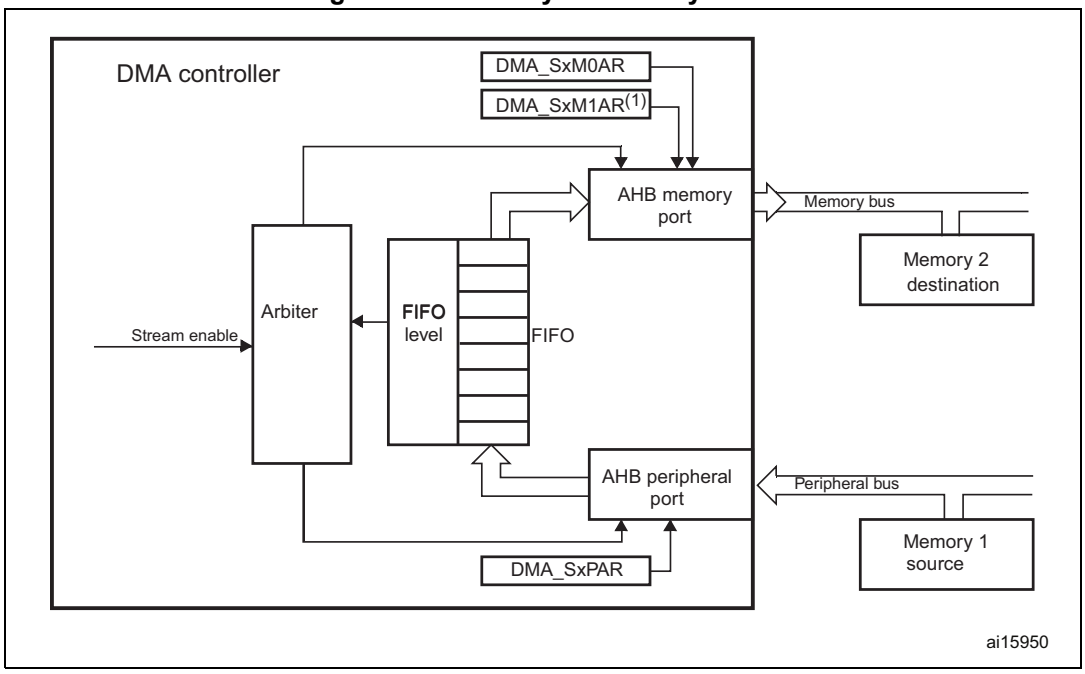
When the stream is enabled by setting the Enable bit (EN) in the DMA_SxCR register, the stream immediately starts to fill the FIFO up to the threshold level. When the threshold level is reached, the FIFO contents are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero or when the EN bit in the DMA_SxCR register is cleared by software.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Note: When memory-to-memory mode is used, the circular and direct modes are not allowed.

Figure 221. Memory-to-memory mode



1. For double-buffer mode.

23.3.9 Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented or kept constant after each transfer depending on the PINC and MINC bits in the DMA_SxCR register.

Disabling the increment mode is useful when the peripheral source or destination data is accessed through a single register.

If the increment mode is enabled, the address of the next transfer is the address of the previous one incremented by 1 (for bytes), 2 (for half-words) or 4 (for words) depending on the data width programmed in the PSIZE or MSIZE bits in the DMA_SxCR register.

In order to optimize the packing operation, it is possible to fix the increment offset size for the peripheral address whatever the size of the data transferred on the AHB peripheral port. The PINCOS bit in the DMA_SxCR register is used to align the increment offset size with the data size on the peripheral AHB port, or on a 32-bit address (the address is then incremented by 4). The PINCOS bit has an impact on the AHB peripheral port only.

If the PINCOS bit is set, the address of the following transfer is the address of the previous one incremented by 4 (automatically aligned on a 32-bit address), whatever the PSIZE value. The AHB memory port, however, is not impacted by this operation.

23.3.10 Circular mode

The circular mode is available to handle circular buffers and continuous data flows (for example ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_SxCR register.

When the circular mode is activated, the number of data items to be transferred is automatically reloaded with the initial value programmed during the stream configuration phase, and the DMA requests continue to be served.

Note: In the circular mode, it is mandatory to respect the following rule in case of a burst mode configured for memory:

DMA_SxNDTR = Multiple of ((Mburst beat) × (Msize)/(Psize)), where:

- *(Mburst beat) = 4, 8 or 16 (depending on the MBURST bits in the DMA_SxCR register)*
- *((Msize)/(Psize)) = 1, 2, 4, 1/2 or 1/4 (Msize and Psize represent the MSIZE and PSIZE bits in the DMA_SxCR register. They are byte dependent)*
- *DMA_SxNDTR = Number of data items to transfer on the AHB peripheral port*

For example: Mburst beat = 8 (INCR8), MSIZE = 00 (byte) and PSIZE = 01 (half-word), in this case: DMA_SxNDTR must be a multiple of (8 × 1/2 = 4).

If this formula is not respected, the DMA behavior and data integrity are not guaranteed.

NDTR must also be a multiple of the Peripheral burst size multiplied by the peripheral data size, otherwise this could result in a bad DMA behavior.

23.3.11 Double-buffer mode

This mode is available for all the DMA1 and DMA2 streams.

The double-buffer mode is enabled by setting the DBM bit in the DMA_SxCR register.

A double-buffer stream works as a regular (single buffer) stream with the difference that it has two memory pointers. When the double-buffer mode is enabled, the circular mode is automatically enabled (CIRC bit in DMA_SxCR is not relevant) and at each end of transaction, the memory pointers are swapped.

In this mode, the DMA controller swaps from one memory target to another at each end of transaction. This allows the software to process one memory area while the second memory area is being filled/used by the DMA transfer. The double-buffer stream can work in both directions (the memory can be either the source or the destination) as described in *Table 293: Source and destination address registers in double-buffer mode (DBM = 1)*.

*Note: In double-buffer mode, it is possible to update the base address for the AHB memory port on-the-fly (*DMA_SxM0AR or DMA_SxM1AR*) when the stream is enabled, by respecting the following conditions:*

- *When the CT bit is 0 in the DMA_SxCR register, the DMA_SxM1AR register can be written. Attempting to write to this register while CT = 1 sets an error flag (TEIF) and the stream is automatically disabled.*
- *When the CT bit is 1 in the DMA_SxCR register, the DMA_SxM0AR register can be written. Attempting to write to this register while CT = 0, sets an error flag (TEIF) and the stream is automatically disabled.*

To avoid any error condition, it is advised to change the base address as soon as the TCIF flag is asserted because, at this point, the targeted memory must have changed from

memory 0 to 1 (or from 1 to 0) depending on the value of CT in the DMA_SxCR register in accordance with one of the two above conditions.

For all the other modes (except the double-buffer mode), the memory address registers are write-protected as soon as the stream is enabled.

Table 293. Source and destination address registers in double-buffer mode (DBM = 1)

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR / DMA_SxM1AR
01	Memory-to-peripheral	DMA_SxM0AR / DMA_SxM1AR	DMA_SxPAR
10	Not allowed(1)		
11	Reserved	-	-

1. When the double-buffer mode is enabled, the circular mode is automatically enabled. Since the memory-to-memory mode is not compatible with the circular mode, when the double-buffer mode is enabled, it is not allowed to configure the memory-to-memory mode.

23.3.12 Programmable data width, packing/unpacking, endianness

The number of data items to be transferred has to be programmed into DMA_SxNDTR (number of data items to transfer bit, NDT) before enabling the stream (except when the flow controller is the peripheral, PFCTRL bit in DMA_SxCR is set).

When using the internal FIFO, the data widths of the source and destination data are programmable through the PSIZE and MSIZE bits in the DMA_SxCR register (can be 8-, 16- or 32-bit).

When PSIZE and MSIZE are not equal:

- The data width of the number of data items to transfer, configured in the DMA_SxNDTR register is equal to the width of the peripheral bus (configured by the PSIZE bits in the DMA_SxCR register). For instance, in case of peripheral-to-memory, memory-to-peripheral or memory-to-memory transfers and if the PSIZE[1:0] bits are configured for half-word, the number of bytes to be transferred is equal to 2 × NDT.
- The DMA controller only copes with little-endian addressing for both source and destination. This is described in *Table 294: Packing/unpacking and endian behavior (bit PINC = MINC = 1)*.

This packing/unpacking procedure may present a risk of data corruption when the operation is interrupted before the data are completely packed/unpacked. So, to ensure data coherence, the stream may be configured to generate burst transfers: in this case, each group of transfers belonging to a burst is indivisible (refer to *Section 23.3.13: Single and burst transfers*).

In direct mode (DMDIS = 0 in the DMA_SxFCR register), the packing/unpacking of data is not possible. In this case, it is not allowed to have different source and destination transfer data widths: both are equal and defined by the PSIZE bits in the DMA_SxCR register. MSIZE bits are not relevant.

Table 294. Packing/unpacking and endian behavior (bit PINC = MINC = 1)

AHB memory port width	AHB peripheral port width	Number of data items to transfer (NDT)	Memory transfer number	Memory port address / byte lane	Peripheral transfer number	Peripheral port address / byte lane	
						PINCOS = 1	PINCOS = 0
8	8	4	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
8	16	2	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
8	32	1	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
16	8	4	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
16	16	2	1 2	0x0 / B1|B0[15:0] 0x2 / B1|B0[15:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
16	32	1	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
32	8	4	1	0x0 / B3|B2|B1|B0[31:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
32	16	2	1	0x0 /B3|B2|B1|B0[31:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
32	32	1	1	0x0 /B3|B2|B1|B0 [31:0]	1	0x0 /B3|B2|B1|B0 [31:0]	0x0 / B3|B2|B1|B0[31:0]

Note: Peripheral port may be the source or the destination (it can also be the memory source in the case of memory-to-memory transfer).

PSIZE, MSIZE and NDT[15:0] must be configured so as to ensure that the last transfer is not incomplete. This can occur when the data width of the peripheral port (PSIZE bits) is

lower than the data width of the memory port (MSIZE bits). This constraint is summarized in the table below.

Table 295. Restriction on NDT versus PSIZE and MSIZE

PSIZE[1:0] of DMA_SxCR	MSIZE[1:0] of DMA_SxCR	NDT[15:0] of DMA_SxNDTR
00 (8-bit)	01 (16-bit)	Must be a multiple of 2.
00 (8-bit)	10 (32-bit)	Must be a multiple of 4.
01 (16-bit)	10 (32-bit)	Must be a multiple of 2.

23.3.13 Single and burst transfers

The DMA controller can generate single transfers or incremental burst transfers of 4, 8 or 16 beats.

The size of the burst is configured by software independently for the two AHB ports by using the MBURST[1:0] and PBURST[1:0] bits in the DMA_SxCR register.

The burst size indicates the number of beats in the burst, not the number of bytes transferred.

To ensure data coherence, each group of transfers that form a burst is indivisible: AHB transfers are locked and the arbiter of the AHB bus matrix does not degrant the DMA master during the sequence of the burst transfer.

Depending on the single or burst configuration, each DMA request initiates a different number of transfers on the AHB peripheral port:

- When the AHB peripheral port is configured for single transfers, each DMA request generates a data transfer of a byte, half-word or word depending on the PSIZE[1:0] bits in the DMA_SxCR register
- When the AHB peripheral port is configured for burst transfers, each DMA request generates 4,8 or 16 beats of byte, half word or word transfers depending on the PBURST[1:0] and PSIZE[1:0] bits in the DMA_SxCR register.

The same as above has to be considered for the AHB memory port considering the MBURST and MSIZE bits.

In direct mode, the stream can only generate single transfers and the MBURST[1:0] and PBURST[1:0] bits are forced by hardware.

The address pointers (DMA_SxPAR or DMA_SxM0AR registers) must be chosen so as to ensure that all transfers within a burst block are aligned on the address boundary equal to the size of the transfer.

The burst configuration has to be selected in order to respect the AHB protocol, where bursts **must not** cross the 1 Kbyte address boundary because the minimum address space that can be allocated to a single slave is 1 Kbyte. This means that the 1-Kbyte address boundary **must not** be crossed by a burst block transfer, otherwise no error is reported by the DMA registers and the result is unpredictable.

23.3.14 FIFO

FIFO structure

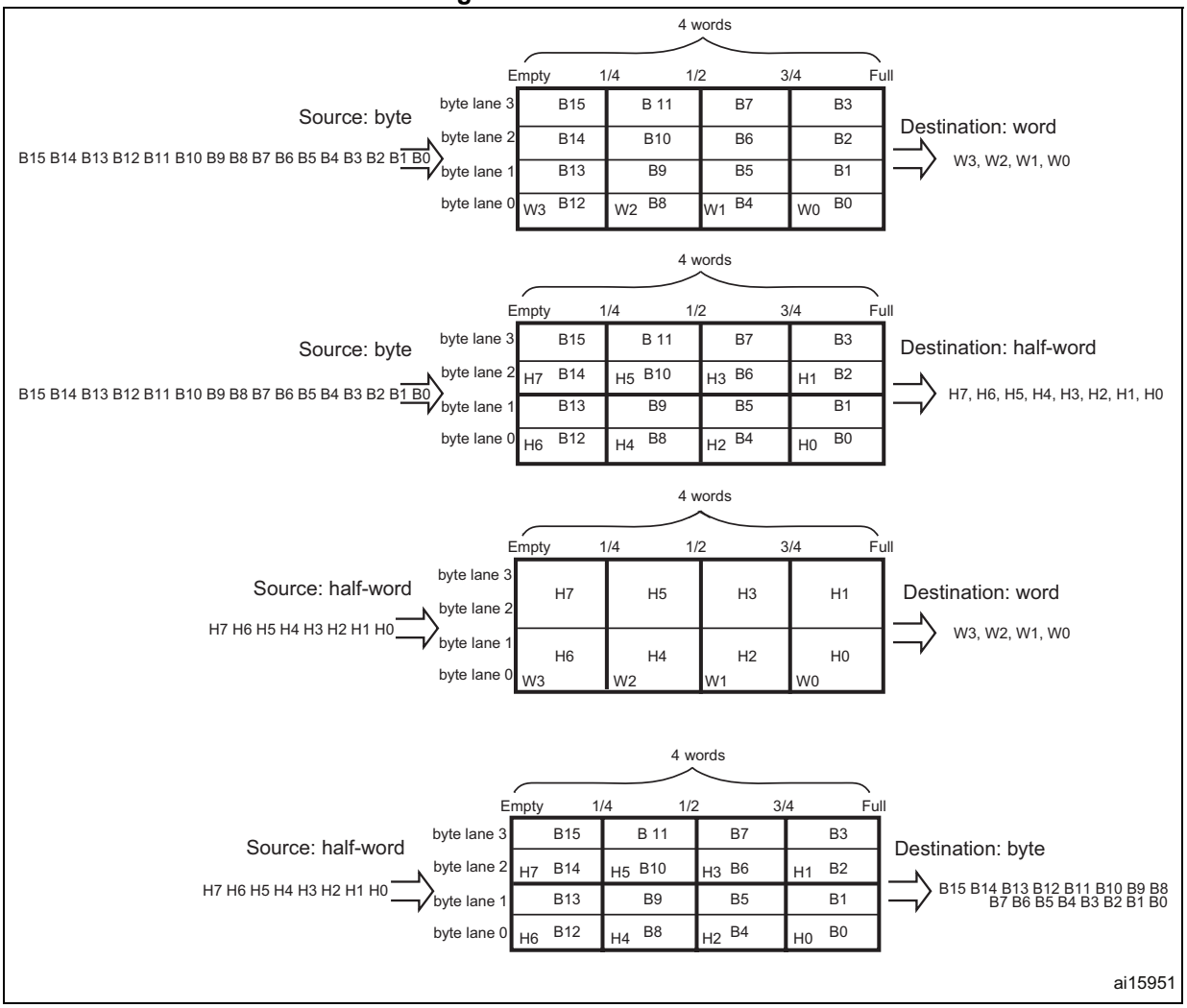
The FIFO is used to temporarily store data coming from the source before transmitting them to the destination.

Each stream has an independent 4-word FIFO and the threshold level is software-configurable between 1/4, 1/2, 3/4 or full.

To enable the use of the FIFO threshold level, the direct mode must be disabled by setting the DMDIS bit in the DMA_SxFCR register.

The structure of the FIFO differs depending on the source and destination data widths, and is described in the figure below.

Figure 222. FIFO structure



FIFO threshold and burst configuration

Caution is required when choosing the FIFO threshold (bits FTH[1:0] of the DMA_SxFCR register) and the size of the memory burst (MBURST[1:0] of the DMA_SxCR register): The content pointed by the FIFO threshold must exactly match an integer number of memory burst transfers. If this is not in the case, a FIFO error (flag FEIFx of the DMA_HISR or DMA_LISR register) is generated when the stream is enabled, then the stream is automatically disabled. The allowed and forbidden configurations are described in the table below. The forbidden configurations are highlighted in gray in the table.

Table 296. FIFO threshold configurations

MSIZE	FIFO level	MBURST = INCR4	MBURST = INCR8	MBURST = INCR16
Byte	1/4	1 burst of 4 beats	Forbidden	Forbidden
	1/2	2 bursts of 4 beats	1 burst of 8 beats	
	3/4	3 bursts of 4 beats	Forbidden	
	Full	4 bursts of 4 beats	2 bursts of 8 beats	1 burst of 16 beats
Half-word	1/4	Forbidden	Forbidden	Forbidden
	1/2	1 burst of 4 beats		
	3/4	Forbidden		
	Full	2 bursts of 4 beats	1 burst of 8 beats	
Word	1/4	Forbidden	Forbidden	
	1/2			
	3/4			
	Full	1 burst of 4 beats		

In all cases, the burst size multiplied by the data size must not exceed the FIFO size (data size can be: 1 (byte), 2 (half-word) or 4 (word)).

Incomplete burst transfer at the end of a DMA transfer may happen if one of the following conditions occurs:

- For the AHB peripheral port configuration: the total number of data items (set in the DMA_SxNDTR register) is not a multiple of the burst size multiplied by the data size.
- For the AHB memory port configuration: the number of remaining data items in the FIFO to be transferred to the memory is not a multiple of the burst size multiplied by the data size.

In such cases, the remaining data to be transferred is managed in single mode by the DMA, even if a burst transaction is requested during the DMA stream configuration.

Note: When burst transfers are requested on the peripheral AHB port and the FIFO is used (DMDIS = 1 in the DMA_SxCR register), it is mandatory to respect the following rule to avoid permanent underrun or overrun conditions, depending on the DMA stream direction:

If (PBURST × PSIZE) = FIFO_SIZE (4 words), FIFO_Threshold = 3/4 is forbidden with PSIZE = 1, 2 or 4 and PBURST = 4, 8 or 16.

This rule ensures that enough FIFO space at a time is free to serve the request from the peripheral.

FIFO flush

The FIFO can be flushed when the stream is disabled by resetting the EN bit in the DMA_SxCR register and when the stream is configured to manage peripheral-to-memory or memory-to-memory transfers. If some data are still present in the FIFO when the stream is disabled, the DMA controller continues transferring the remaining data to the destination (even though stream is effectively disabled). When this flush is completed, the transfer complete status bit (TCIFx) in the DMA_LISR or DMA_HISR register is set.

The remaining data counter DMA_SxNDTR keeps the value in this case to indicate how many data items are currently available in the destination memory.

Note that during the FIFO flush operation, if the number of remaining data items in the FIFO to be transferred to memory (in bytes) is less than the memory data width (for example 2 bytes in FIFO while MSIZE is configured to word), data is sent with the data width set in the MSIZE bit in the DMA_SxCR register. This means that memory is written with an undesired value. The software may read the DMA_SxNDTR register to determine the memory area that contains the good data (start address and last address).

If the number of remaining data items in the FIFO is lower than a burst size (if the MBURST bits in DMA_SxCR register are set to configure the stream to manage burst on the AHB memory port), single transactions are generated to complete the FIFO flush.

Direct mode

By default, the FIFO operates in direct mode (DMDIS bit in the DMA_SxFCR is reset) and the FIFO threshold level is not used. This mode is useful when the system requires an immediate and single transfer to or from the memory after each DMA request.

When the DMA is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.

To avoid saturating the FIFO, it is recommended to configure the corresponding stream with a high priority.

This mode is restricted to transfers where:

- The source and destination transfer widths are equal and both defined by the PSIZE[1:0] bits in DMA_SxCR (MSIZE[1:0] bits are not relevant)
- Burst transfers are not possible (PBURST[1:0] and MBURST[1:0] bits in DMA_SxCR are don’t care)

Direct mode must not be used when implementing memory-to-memory transfers.

23.3.15 DMA transfer completion

Different events can generate an end of transfer by setting the TCIFx bit in the DMA_LISR or DMA_HISR status register:

- In DMA flow controller mode:
  - The DMA_SxNDTR counter has reached zero in the memory-to-peripheral mode.
  - The stream is disabled before the end of transfer (by clearing the EN bit in the DMA_SxCR register) and (when transfers are peripheral-to-memory or memory-

to-memory) all the remaining data have been flushed from the FIFO into the memory.

- In peripheral flow controller mode:
  - The last external burst or single request has been generated from the peripheral and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory.
  - The stream is disabled by software, and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory.

Note: The transfer completion is dependent on the remaining data in FIFO to be transferred into memory only in the case of peripheral-to-memory mode. This condition is not applicable in memory-to-peripheral mode.

If the stream is configured in non-circular mode, after the end of the transfer (that is when the number of data to be transferred reaches zero), the DMA is stopped (EN bit in DMA_SxCR register is cleared by Hardware) and no DMA request is served unless the software reprograms the stream and re-enables it (by setting the EN bit in the DMA_SxCR register).

23.3.16 DMA transfer suspension

At any time, a DMA transfer can be suspended to be restarted later on or to be definitively disabled before the end of the DMA transfer.

There are two cases:

- The stream disables the transfer with no later-on restart from the point where it was stopped. There is no particular action to do, except to clear the EN bit in the DMA_SxCR register to disable the stream. The stream may take time to be disabled (ongoing transfer is completed first). The transfer complete interrupt flag (TCIF in the DMA_LISR or DMA_HISR register) is set in order to indicate the end of transfer. The value of the EN bit in DMA_SxCR is now 0 to confirm the stream interruption. The DMA_SxNDTR register contains the number of remaining data items at the moment when the stream was stopped so that the software can determine how many data items have been transferred before the stream was interrupted.
- The stream suspends the transfer before the number of remaining data items to be transferred in the DMA_SxNDTR register reaches 0. The aim is to restart the transfer later by re-enabling the stream. In order to restart from the point where the transfer was stopped, the software has to read the DMA_SxNDTR register after disabling the stream by writing the EN bit in DMA_SxCR register (and then checking that it is at 0) to know the number of data items already collected. Then:
  - The peripheral and/or memory addresses have to be updated in order to adjust the address pointers.
  - The SxNDTR register has to be updated with the remaining number of data items to be transferred (the value read when the stream was disabled).
  - The stream may then be re-enabled to restart the transfer from the point it was stopped.

Note: A transfer complete interrupt flag (TCIF in DMA_LISR or DMA_HISR) is set to indicate the end of transfer due to the stream interruption.

23.3.17 Flow controller

The entity that controls the number of data to be transferred is known as the flow controller. This flow controller is configured independently for each stream using the PFCTRL bit in the DMA_SxCR register.

The flow controller can be:

- The DMA controller: in this case, the number of data items to be transferred is programmed by software into the DMA_SxNDTR register before the DMA stream is enabled.
- The peripheral source or destination: this is the case when the number of data items to be transferred is unknown. The peripheral indicates by hardware to the DMA controller when the last data are being transferred. This feature is only supported for peripherals that are able to signal the end of the transfer.

When the peripheral flow controller is used for a given stream, the value written into the DMA_SxNDTR has no effect on the DMA transfer. Actually, whatever the value written, it is forced by hardware to 0xFFFF as soon as the stream is enabled, to respect the following schemes:

- Anticipated stream interruption: EN bit in DMA_SxCR register is reset to 0 by the software to stop the stream before the last data hardware signal (single or burst) is sent by the peripheral. In such a case, the stream is switched off and the FIFO flush is triggered in the case of a peripheral-to-memory DMA transfer. The TCIFx flag of the corresponding stream is set in the status register to indicate the DMA completion. To know the number of data items transferred during the DMA transfer, read the DMA_SxNDTR register and apply the following formula:
  – Number_of_data_transferred = 0xFFFF – DMA_SxNDTR
- Normal stream interruption due to the reception of a last data hardware signal: the stream is automatically interrupted when the peripheral requests the last transfer (single or burst) and when this transfer is complete. the TCIFx flag of the corresponding stream is set in the status register to indicate the DMA transfer completion. To know the number of data items transferred, read the DMA_SxNDTR register and apply the same formula as above.
- The DMA_SxNDTR register reaches 0: the TCIFx flag of the corresponding stream is set in the status register to indicate the forced DMA transfer completion. The stream is automatically switched off even though the last data hardware signal (single or burst) has not been yet asserted. The already transferred data is not lost. This means that a maximum of 65535 data items can be managed by the DMA in a single transaction, even in peripheral flow control mode.

Note: When configured in memory-to-memory mode, the DMA is always the flow controller and the PFCTRL bit is forced to 0 by hardware.

The circular mode is forbidden in the peripheral flow controller mode.

23.3.18 Summary of the possible DMA configurations

The table below summarizes the different possible DMA configurations. The forbidden configurations are highlighted in gray in the table.

Table 297. Possible DMA configurations

DMA transfer mode	Source	Destination	Flow controller	Circular mode	Transfer type	Direct mode	Double-buffer mode
Peripheral-to-memory	AHB peripheral port	AHB memory port	DMA	Possible	single	Possible	Possible
					burst	Forbidden	
			Peripheral	Forbidden	single	Possible	Forbidden
					burst	Forbidden	
Memory-to-peripheral	AHB memory port	AHB peripheral port	DMA	Possible	single	Possible	Possible
					burst	Forbidden	
			Peripheral	Forbidden	single	Possible	Forbidden
					burst	Forbidden	
Memory-to-memory	AHB peripheral port	AHB memory port	DMA only	Forbidden	single	Forbidden	Forbidden
					burst		

23.3.19 Stream configuration procedure

The following sequence must be followed to configure a DMA stream x (where x is the stream number):

1. If the stream is enabled, disable it by resetting the EN bit in the DMA_SxCR register, then read this bit in order to confirm that there is no ongoing stream operation. Writing this bit to 0 is not immediately effective since it is actually written to 0 once all the current transfers are finished. When the EN bit is read as 0, this means that the stream is ready to be configured. It is therefore necessary to wait for the EN bit to be cleared before starting any stream configuration. All the stream dedicated bits set in the status register (DMA_LISR and DMA_HISR) from the previous data block DMA transfer must be cleared before the stream can be re-enabled.
2. Set the peripheral port register address in the DMA_SxPAR register. The data is moved from/ to this address to/ from the peripheral port after the peripheral event.
3. Set the memory address in the DMA_SxMA0R register (and in the DMA_SxMA1R register in the case of a double-buffer mode). The data is written to or read from this memory after the peripheral event.
4. Configure the total number of data items to be transferred in the DMA_SxNDTR register. After each peripheral event or each beat of the burst, this value is decremented.
5. Use DMAMUX1 to route a DMA request line to the DMA channel.
6. If the peripheral is intended to be the flow controller and if it supports this feature, set the PFCTRL bit in the DMA_SxCR register.
7. Configure the stream priority using the PL[1:0] bits in the DMA_SxCR register.
8. Configure the FIFO usage (enable or disable, threshold in transmission and reception)

9. Configure the data transfer direction, peripheral and memory incremented/fixed mode, single or burst transactions, peripheral and memory data widths, circular mode, double-buffer mode and interrupts after half and/or full transfer, and/or errors in the DMA_SxCR register.
10. Activate the stream by setting the EN bit in the DMA_SxCR register.

As soon as the stream is enabled, it can serve any DMA request from the peripheral connected to the stream.

Once half the data have been transferred on the AHB destination port, the half-transfer flag (HTIF) is set and an interrupt is generated if the half-transfer interrupt enable bit (HTIE) is set. At the end of the transfer, the transfer complete flag (TCIF) is set and an interrupt is generated if the transfer complete interrupt enable bit (TCIE) is set.

Warning: To switch off a peripheral connected to a DMA stream request, it is mandatory to, first, switch off the DMA stream to which the peripheral is connected, then to wait for EN bit = 0. Only then can the peripheral be safely disabled.

23.3.20 Error management

The DMA controller can detect the following errors:

- **Transfer error**: the transfer error interrupt flag (TEIFx) is set when:
  - a bus error occurs during a DMA read or a write access
  - a write access is requested by software on a memory address register in double-buffer mode whereas the stream is enabled and the current target memory is the one impacted by the write into the memory address register (refer to *Section 23.3.11: Double-buffer mode*)
- **FIFO error**: the FIFO error interrupt flag (FEIFx) is set if:
  - a FIFO underrun condition is detected
  - a FIFO overrun condition is detected (no detection in memory-to-memory mode because requests and transfers are internally managed by the DMA)
  - the stream is enabled while the FIFO threshold level is not compatible with the size of the memory burst (refer to *Table 296: FIFO threshold configurations*)
- **Direct mode error**: the direct mode error interrupt flag (DMEIFx) can only be set in the peripheral-to-memory mode while operating in direct mode and when the MINC bit in the DMA_SxCR register is cleared. This flag is set when a DMA request occurs while the previous data have not yet been fully transferred into the memory (because the memory bus was not granted). In this case, the flag indicates that two data items were transferred successively to the same destination address, which could be an issue if the destination is not able to manage this situation

In direct mode, the FIFO error flag can also be set under the following conditions:

- In the peripheral-to-memory mode, the FIFO can be saturated (overrun) if the memory bus is not granted for several peripheral requests.
- In the memory-to-peripheral mode, an underrun condition may occur if the memory bus has not been granted before a peripheral request occurs.

If the TEIFx or the FEIFx flag is set due to incompatibility between burst size and FIFO threshold level, the faulty stream is automatically disabled through a hardware clear of its EN bit in the corresponding stream configuration register (DMA_SxCR).

If the DMEIFx or the FEIFx flag is set due to an overrun or underrun condition, the faulty stream is not automatically disabled and it is up to the software to disable or not the stream by resetting the EN bit in the DMA_SxCR register. This is because there is no data loss when this kind of errors occur.

When the stream's error interrupt flag (TEIF, FEIF, DMEIF) in the DMA_LISR or DMA_HISR register is set, an interrupt is generated if the corresponding interrupt enable bit (TEIE, FEIE, DMIE) in the DMA_SxCR or DMA_SxFCR register is set.

Note: When a FIFO overrun or underrun condition occurs, the data is not lost because the peripheral request is not acknowledged by the stream until the overrun or underrun condition is cleared. If this acknowledge takes too much time, the peripheral itself may detect an overrun or underrun condition of its internal buffer and data might be lost.

23.4 DMA interrupts

For each DMA stream, an interrupt can be produced on the following events:

- Half-transfer reached
- Transfer complete
- Transfer error
- FIFO error (overrun, underrun or FIFO level error)
- Direct mode error

Separate interrupt enable control bits are available for flexibility as shown in the table below.

Table 298. DMA interrupt requests

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE
FIFO overrun/underrun	FEIF	FEIE
Direct mode error	DMEIF	DMEIE

Note: Before setting an enable control bit EN = 1, the corresponding event flag must be cleared, otherwise an interrupt is immediately generated.

23.5 DMA registers

The DMA registers have to be accessed by words (32 bits).

23.5.1 DMA memory map

Table 299. DMA register memory map

Offset	Register name
0x000	*DMA low interrupt status register (DMA_LISR)*
0x004	*DMA high interrupt status register (DMA_HISR)*
0x008	*DMA low interrupt flag clear register (DMA_LIFCR)*
0x00C	*DMA high interrupt flag clear register (DMA_HIFCR)*
0x010 + 0x018 * x, (x = 0 to 7)	*DMA stream x configuration register (DMA_SxCR)*
0x014 + 0x018 * x, (x = 0 to 7)	*DMA stream x number of data register (DMA_SxNDTR)*
0x018 + 0x018 * x, (x = 0 to 7)	*DMA stream x peripheral address register (DMA_SxPAR)*
0x01C + 0x018 * x, (x = 0 to 7)	*DMA stream x memory 0 address register (DMA_SxM0AR)*
0x020 + 0x018 * x, (x = 0 to 7)	*DMA stream x memory 1 address register (DMA_SxM1AR)*
0x024 + 0x018 * x, (x = 0 to 7)	*DMA stream x FIFO control register (DMA_SxFCR)*

23.5.2 DMA low interrupt status register (DMA_LISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TCIF3	HTIF3	TEIF3	DMEIF3	Res.	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Res.	FEIF2
				r	r	r	r		r	r	r	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TCIF1	HTIF1	TEIF1	DMEIF1	Res.	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Res.	FEIF0
				r	r	r	r		r	r	r	r	r		r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIF[3:0]**: stream x transfer complete interrupt flag (x = 3 to 0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: no transfer complete event on stream x

1: a transfer complete event occurred on stream x

Bits 26, 20, 10, 4 **HTIF[3:0]**: stream x half transfer interrupt flag (x = 3 to 0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: no half transfer event on stream x

1: a half transfer event occurred on stream x

Bits 25, 19, 9, 3 **TEIF[3:0]**: stream x transfer error interrupt flag (x = 3 to 0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.
0: no transfer error on stream x
1: a transfer error occurred on stream x

Bits 24, 18, 8, 2 **DMEIF[3:0]**: stream x direct mode error interrupt flag (x = 3 to 0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.
0: No direct mode error on stream x
1: a direct mode error occurred on stream x

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIF[3:0]**: stream x FIFO error interrupt flag (x = 3 to 0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.
0: no FIFO error event on stream x
1: a FIFO error event occurred on stream x

23.5.3 DMA high interrupt status register (DMA_HISR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TCIF7	HTIF7	TEIF7	DMEIF7	Res.	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Res.	FEIF6
				r	r	r	r		r	r	r	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TCIF5	HTIF5	TEIF5	DMEIF5	Res.	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Res.	FEIF4
				r	r	r	r		r	r	r	r	r		r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIF[7:4]**: stream x transfer complete interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.
0: no transfer complete event on stream x
1: a transfer complete event occurred on stream x

Bits 26, 20, 10, 4 **HTIF[7:4]**: stream x half transfer interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.
0: no half transfer event on stream x
1: a half transfer event occurred on stream x

Bits 25, 19, 9, 3 **TEIF[7:4]**: stream x transfer error interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.
0: no transfer error on stream x
1: a transfer error occurred on stream x

Bits 24, 18, 8, 2 **DMEIF[7:4]**: stream x direct mode error interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.
0: no direct mode error on stream x
1: a direct mode error occurred on stream x

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIF[7:4]**: stream x FIFO error interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.
0: no FIFO error event on stream x
1: a FIFO error event occurred on stream x

23.5.4 DMA low interrupt flag clear register (DMA_LIFCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTCIF3	CHTIF3	CTEIF3	CDMEIF3	Res.	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Res.	CFEIF2
				w	w	w	w		w	w	w	w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Res.	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Res.	CFEIF0
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIF[3:0]**: stream x clear transfer complete interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_LISR register.

Bits 26, 20, 10, 4 **CHTIF[3:0]**: stream x clear half transfer interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_LISR register

Bits 25, 19, 9, 3 **CTEIF[3:0]**: Stream x clear transfer error interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_LISR register.

Bits 24, 18, 8, 2 **CDMEIF[3:0]**: stream x clear direct mode error interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_LISR register.

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIF[3:0]**: stream x clear FIFO error interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_LISR register.

23.5.5 DMA high interrupt flag clear register (DMA_HIFCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Res.	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Res.	CFEIF6
				w	w	w	w		w	w	w	w	w		w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Res.	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Res.	CFEIF4
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIF[7:4]**: stream x clear transfer complete interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_HISR register.

Bits 26, 20, 10, 4 **CHTIF[7:4]**: stream x clear half transfer interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_HISR register.

Bits 25, 19, 9, 3 **CTEIF[7:4]**: stream x clear transfer error interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_HISR register.

Bits 24, 18, 8, 2 **CDMEIF[7:4]**: stream x clear direct mode error interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_HISR register.

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIF[7:4]**: stream x clear FIFO error interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_HISR register.

23.5.6 DMA stream x configuration register (DMA_SxCR)

This register is used to configure the concerned stream.

Address offset: 0x010 + 0x018 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST[1:0]		PBURST[1:0]		TR BUFF	CT	DBM	PL[1:0]	
							rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:23 **MBURST[1:0]**: memory burst transfer configuration
These bits are set and cleared by software.
00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN = 0.
In direct mode, these bits are forced to 0x0 by hardware as soon as bit EN = 1.

Bits 22:21 **PBURST[1:0]**: peripheral burst transfer configuration
These bits are set and cleared by software.
00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN = 0.
In direct mode, these bits are forced to 0x0 by hardware.

Bit 20 **TRBUFF**: Enable the DMA to handle bufferable transfers.
0: bufferable transfers not enabled
1: bufferable transfers enabled
Note: This bit must be set to 1 if the DMA stream manages UART transfers.

Bit 19 **CT**: current target (only in double-buffer mode)
This bit is set and cleared by hardware. It can also be written by software.
0: current target memory is Memory 0 (addressed by the DMA_SxM0AR pointer)
1: current target memory is Memory 1 (addressed by the DMA_SxM1AR pointer)
This bit can be written only if EN = 0 to indicate the target memory area of the first transfer.
Once the stream is enabled, this bit operates as a status flag indicating which memory area is the current target.

Bit 18 **DBM**: double-buffer mode
This bit is set and cleared by software.
0: no buffer switching at the end of transfer
1: memory target switched at the end of the DMA transfer
This bit is protected and can be written only if EN = 0.

Bits 17:16 **PL[1:0]**: priority level
These bits are set and cleared by software.
00: low
01: medium
10: high
11: very high
These bits are protected and can be written only if EN = 0.

Bit 15 **PINCOS**: peripheral increment offset size
This bit is set and cleared by software
0: The offset size for the peripheral address calculation is linked to the PSIZE
1: The offset size for the peripheral address calculation is fixed to 4 (32-bit alignment).
This bit has no meaning if bit PINC = 0.
This bit is protected and can be written only if EN = 0.
This bit is forced low by hardware when the stream is enabled (EN = 1) if the direct mode is selected or if PBURST are different from 00.

Bits 14:13 **MSIZE[1:0]**: memory data size
These bits are set and cleared by software.
00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved
These bits are protected and can be written only if EN = 0.
In direct mode, MSIZE is forced by hardware to the same value as PSIZE as soon as EN = 1.

Bits 12:11 **PSIZE[1:0]**: peripheral data size
These bits are set and cleared by software.
00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved
These bits are protected and can be written only if EN = 0.

Bit 10 **MINC**: memory increment mode
This bit is set and cleared by software.
0: memory address pointer is fixed
1: memory address pointer is incremented after each data transfer (increment is done according to MSIZE)
This bit is protected and can be written only if EN = 0.

Bit 9 **PINC**: peripheral increment mode
This bit is set and cleared by software.
0: peripheral address pointer fixed
1: peripheral address pointer incremented after each data transfer (increment done according to PSIZE)
This bit is protected and can be written only if EN = 0.

Bit 8 **CIRC**: circular mode
This bit is set and cleared by software and can be cleared by hardware.
0: circular mode disabled
1: circular mode enabled
When the peripheral is the flow controller (bit PFCTRL = 1) and the stream is enabled (EN = 1), then this bit is automatically forced by hardware to 0.
It is automatically forced by hardware to 1 if the DBM bit is set, as soon as the stream is enabled (EN = 1).

Bits 7:6 **DIR[1:0]**: data transfer direction
These bits are set and cleared by software.
00: peripheral-to-memory
01: memory-to-peripheral
10: memory-to-memory
11: reserved
These bits are protected and can be written only if EN = 0.

Bit 5 **PFCTRL**: peripheral flow controller
This bit is set and cleared by software.
0: DMA is the flow controller.
1: The peripheral is the flow controller.
This bit is protected and can be written only if EN = 0.
When the memory-to-memory mode is selected (bits DIR[1:0]=10), then this bit is automatically forced to 0 by hardware.

Bit 4 **TCIE**: transfer complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 3 **HTIE**: half transfer interrupt enable
This bit is set and cleared by software.
0: HT interrupt disabled
1: HT interrupt enabled

Bit 2 **TEIE**: transfer error interrupt enable
This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 1 **DMEIE**: direct mode error interrupt enable
This bit is set and cleared by software.
0: DME interrupt disabled
1: DME interrupt enabled

Bit 0 **EN**: stream enable / flag stream ready when read low
This bit is set and cleared by software.
0: stream disabled
1: stream enabled
This bit may be cleared by hardware:
– on a DMA end of transfer (stream ready to be configured)
– if a transfer error occurs on the AHB master buses
– when the FIFO threshold on memory AHB port is not compatible with the size of the burst
When this bit is read as 0, the software is allowed to program the configuration and FIFO bits registers. It is forbidden to write these registers when the EN bit is read as 1.

Note: Before setting EN bit to 1 to start a new transfer, the event flags corresponding to the stream in DMA_LISR or DMA_HISR register must be cleared.

23.5.7 DMA stream x number of data register (DMA_SxNDTR)

Address offset: 0x014 + 0x018 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **NDT[15:0]**: number of data items to transfer (0 up to 65535)

This register can be written only when the stream is disabled. When the stream is enabled, this register is read-only, indicating the remaining data items to be transmitted. This register decrements after each DMA transfer.

Once the transfer is completed, this register can either stay at zero (when the stream is in normal mode) or be reloaded automatically with the previously programmed value in the following cases:

- when the stream is configured in circular mode.
- when the stream is enabled again by setting EN bit to 1.

If the value of this register is zero, no transaction can be served even if the stream is enabled.

23.5.8 DMA stream x peripheral address register (DMA_SxPAR)

Address offset: 0x018 + 0x018 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PAR[31:0]**: peripheral address

Base address of the peripheral data register from/to which the data is read/written.

These bits are write-protected and can be written only when bit EN = 0 in DMA_SxCR.

23.5.9 DMA stream x memory 0 address register (DMA_SxM0AR)

Address offset: 0x01C + 0x018 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M0A[31:0]**: memory 0 address

Base address of memory area 0 from/to which the data is read/written.

These bits are write-protected. They can be written only if:

- the stream is disabled (EN = 0 in DMA_SxCR) or
- the stream is enabled (EN = 1 in DMA_SxCR) and CT = 1 in DMA_SxCR (in double-buffer mode).

23.5.10 DMA stream x memory 1 address register (DMA_SxM1AR)

Address offset: 0x020 + 0x018 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M1A[31:0]**: memory 1 address (used in case of double-buffer mode)
Base address of memory area 1 from/to which the data is read/written.
This register is used only for the double-buffer mode.
These bits are write-protected. They can be written only if:
– the stream is disabled (EN = 0 in DMA_SxCR) or
– the stream is enabled (EN = 1 in DMA_SxCR) and bit CT = 0 in DMA_SxCR.

23.5.11 DMA stream x FIFO control register (DMA_SxFCR)

Address offset: 0x024 + 0x018 * x, (x = 0 to 7)

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEIE	Res.	FS[2:0]			DMDIS	FTH[1:0]	
								rw		r	r	r	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **FEIE**: FIFO error interrupt enable
This bit is set and cleared by software.
0: FE interrupt disabled
1: FE interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bits 5:3 **FS[2:0]**: FIFO status
These bits are read-only.
000: 0 < fifo_level < 1/4
001: 1/4 ≤ fifo_level < 1/2
010: 1/2 ≤ fifo_level < 3/4
011: 3/4 ≤ fifo_level < full
100: FIFO is empty
101: FIFO is full
others: no meaning
These bits are not relevant in the direct mode (DMDIS = 0).

Bit 2 **DMDIS**: direct mode disable

This bit is set and cleared by software. It can be set by hardware.

0: direct mode enabled

1: direct mode disabled

This bit is protected and can be written only if EN = 0.

This bit is set by hardware if the memory-to-memory mode is selected (DIR bit in DMA_SxCR are 10) and the EN = 1 in DMA_SxCR because the direct mode is not allowed in the memory-to-memory configuration.

Bits 1:0 **FTH[1:0]**: FIFO threshold selection

These bits are set and cleared by software.

00: 1/4 full FIFO

01: 1/2 full FIFO

10: 3/4 full FIFO

11: full FIFO

These bits are not used in the direct mode when the DMIS = 0.

These bits are protected and can be written only if EN = 0.

24 DMA request multiplexer (DMAMUX)

24.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until it is served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is deasserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals and the DMA controllers of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in *Section 5.2.7.1: DMAMUX instantiation*.

The assignment of DMAMUX request multiplexer inputs to the DMA request lines from peripherals and to the DMAMUX request generator outputs, the assignment of DMAMUX request multiplexer outputs to DMA controller channels, and the assignment of DMAMUX synchronizations and trigger inputs to internal and external signals depend on the product implementation, and are detailed in*Section 5.2.7.2: DMAMUX mapping*.

24.2 DMAMUX main features

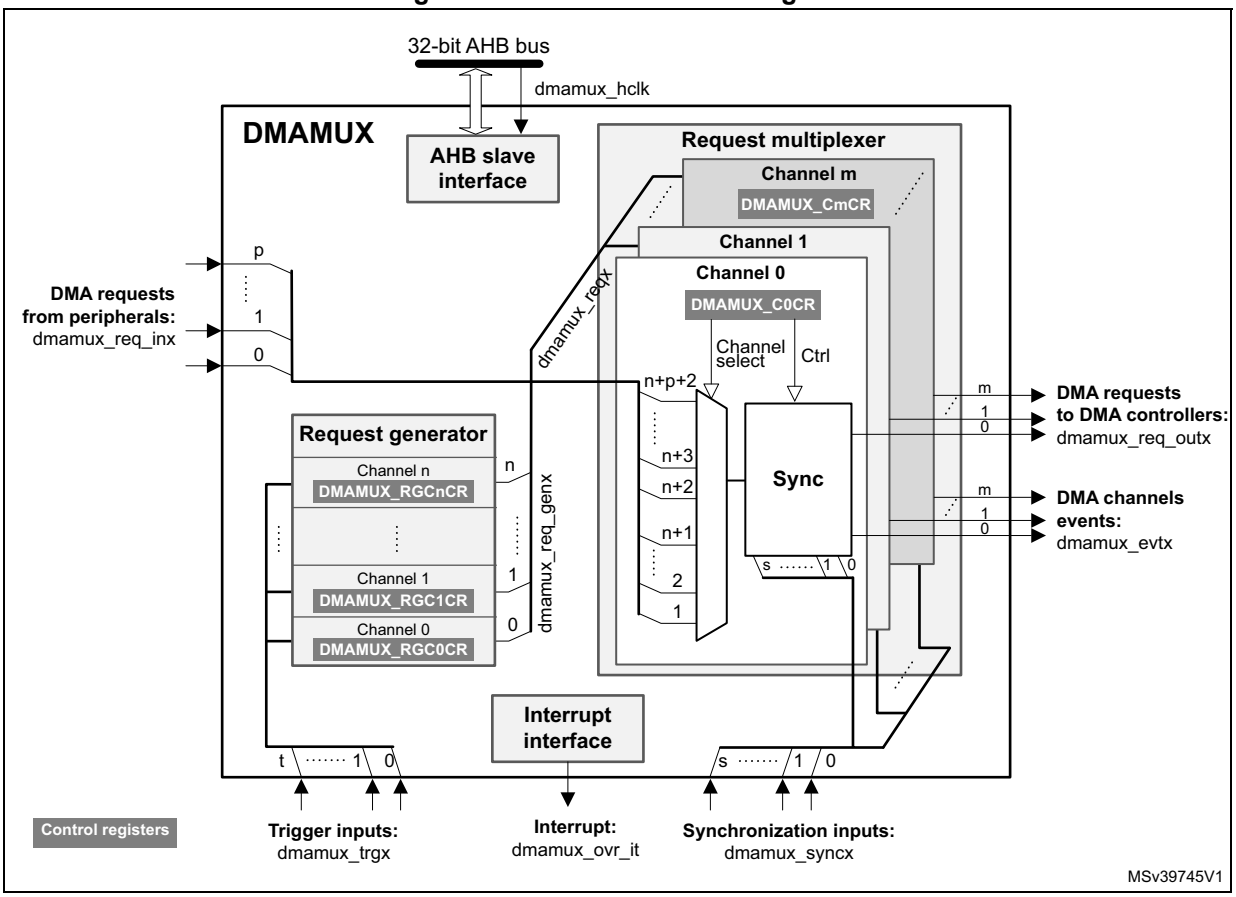
- Up to 16-channel programmable DMA request line multiplexer output
- 4-channel DMA request generator
- 21 trigger inputs to DMA request generator
- 21 synchronization inputs
- Per DMA request generator channel:
  - DMA request trigger input selector
  - DMA request counter
  - Event overrun flag for selected DMA request trigger input
- Per DMA request line multiplexer channel output:
  - 115 input DMA request lines from peripherals
  - One DMA request line output
  - Synchronization input selector
  - DMA request counter
  - Event overrun flag for selected synchronization input
  - One event output, for DMA request chaining

24.3 DMAMUX functional description

24.3.1 DMAMUX block diagram

Figure 223 shows the DMAMUX block diagram.

Figure 223. DMAMUX block diagram



DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (dmamux_reqx) from peripherals (dmamux_req_inx) and from channels of the DMAMUX request generator sub-block (dmamux_req_genx)
- DMAMUX request outputs to channels of DMA controllers (dmamux_req_outx)
- Internal or external signals to DMA request trigger inputs (dmamux_trgx)
- Internal or external signals to synchronization inputs (dmamux_syncx)

24.3.2 DMAMUX signals

Table 300 lists the DMAMUX signals.

Table 300. DMAMUX signals

Signal name	Description
dmamux_hclk	DMAMUX AHB clock
dmamux_req_inx	DMAMUX DMA request line inputs from peripherals
dmamux_trgx	DMAMUX DMA request triggers inputs (to request generator sub-block)
dmamux_req_genx	DMAMUX request generator sub-block channels outputs
dmamux_reqx	DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels)
dmamux_syncx	DMAMUX synchronization inputs (to request multiplexer sub-block)
dmamux_req_outx	DMAMUX requests outputs (to DMA controllers)
dmamux_evtx	DMAMUX events outputs
dmamux_ovr_it	DMAMUX overrun interrupts

24.3.3 DMAMUX channels

A DMAMUX channel is a DMAMUX request multiplexer channel that may include, depending on the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to one single channel of DMA controller(s).

Channel configuration procedure

Follow the sequence below to configure both a DMAMUX x channel and the related DMA channel y:

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

24.3.4 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced either from the peripherals or from the DMAMUX request generator.

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the DMAREQ_ID field in the DMAMUX_CxCR register.

Note: The null value in the field DMAREQ_ID corresponds to no DMA request line selected.

Caution: A same non-null DMAREQ_ID can be assigned to two different channels only if the application ensures that these channels are not requested to be served at the same time. In other words, if two different channels receive the same asserted hardware request at the same time, an unpredictable DMA hardware behavior occurs.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

Synchronization mode and channel event generation

Each DMAMUX request line multiplexer channel x can be individually synchronized by setting the synchronization enable (SE) bit in the DMAMUX_CxCR register.

DMAMUX has multiple synchronization inputs. The synchronization inputs are connected in parallel to all the channels of the request multiplexer.

The synchronization input is selected via the SYNC_ID field in the DMAMUX_CxCR register of a given channel x.

When a channel is in this synchronization mode, the selected input DMA request line is propagated to the multiplexer channel output once a programmable rising/falling edge is detected on the selected input synchronization signal, through the SPOL[1:0] field of the DMAMUX_CxCR register.

Additionally, there is a programmable DMA request counter, internally to the DMAMUX request multiplexer, which may be used for the channel request output generation and also possibly for an event generation. An event generation on the channel x output is enabled through the EGE bit (event generation enable) of the DMAMUX_CxCR register.

As shown in *Figure 225*, upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

Note: If a synchronization event occurs while there is no pending selected input DMA request line, it is discarded. The following asserted input request line is not connected to the DMAMUX multiplexer channel output until a synchronization event occurs again.

From this point on, each time the connected DMAMUX request is served by the DMA controller (a served request is deasserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in NBREQ field of the DMAMUX_CxCR register and the input DMA request line is disconnected from the multiplexer channel x output.

Thus, the number of DMA requests transferred to the multiplexer channel x output following a detected synchronization event, is equal to the value in NBREQ field, plus one.

Note: The NBREQ field value must only be written by software when both synchronization enable bit SE and event generation enable EGE bit of the corresponding multiplexer channel x are disabled.

Figure 224. Synchronization mode of the DMAMUX request line multiplexer channel

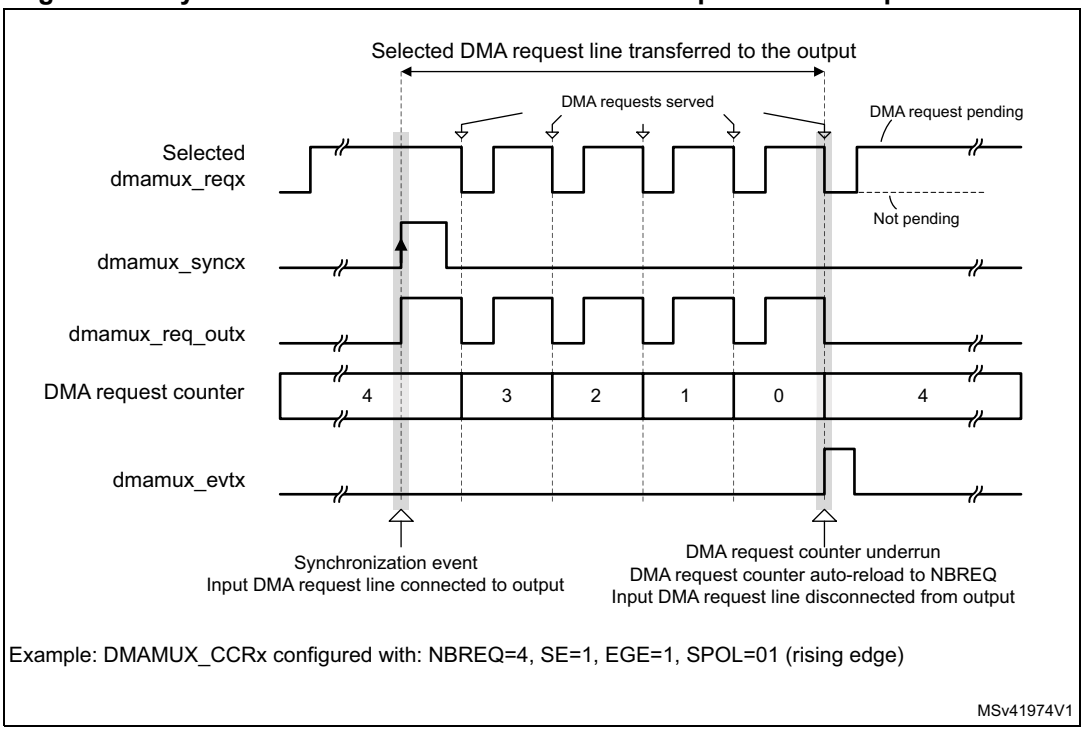
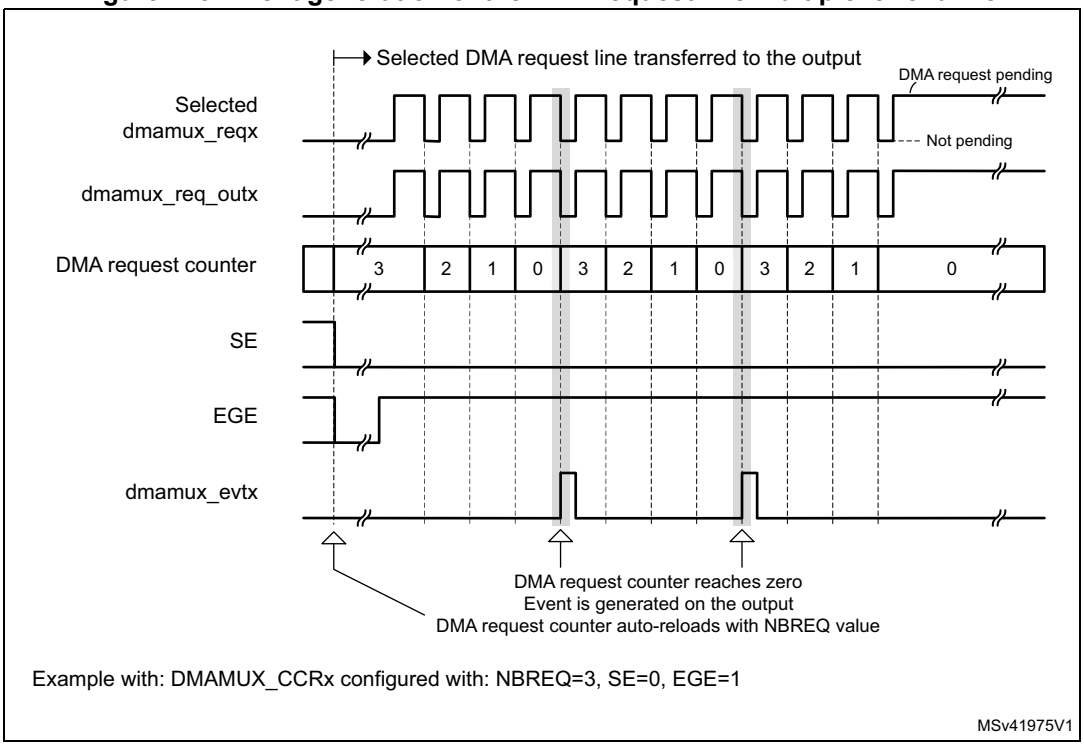


Figure 225. Event generation of the DMA request line multiplexer channel



If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in *Figure 224* and *Figure 225*.

Note: If EGE is enabled and NBREQ = 0, an event is generated after each served DMA request.

Note: A synchronization event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_CxCR register, the synchronization events are masked during three AHB clock cycles.

Synchronization overrun and interrupt

If a new synchronization event occurs before the request counter underrun (the internal request counter programmed via the NBREQ field of the DMAMUX_CxCR register), the synchronization overrun flag bit SOFx is set in the DMAMUX_CSR status register.

Note: The request multiplexer channel x synchronization must be disabled (DMAMUX_CxCR.SE = 0) at the completion of the use of the related channel of the DMA controller. Else, upon a new detected synchronization event, there is a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.

The overrun flag SOFx is reset by setting the associated clear synchronization overrun flag bit CSOFx in the DMAMUX_CFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX_CxCR register.

24.3.5 DMAMUX request generator

The DMAMUX request generator produces DMA requests following trigger events on its DMA request trigger inputs.

The DMAMUX request generator has multiple channels. DMA request trigger inputs are connected in parallel to all channels.

The outputs of DMAMUX request generator channels are inputs to the DMAMUX request line multiplexer.

Each DMAMUX request generator channel x has an enable bit GE (generator enable) in the corresponding DMAMUX_RGxCR register.

The DMA request trigger input for the DMAMUX request generator channel x is selected through the SIG_ID (trigger signal ID) field in the corresponding DMAMUX_RGxCR register.

Trigger events on a DMA request trigger input can be rising edge, falling edge or either edge. The active edge is selected through the GPOL (generator polarity) field in the corresponding DMAMUX_RGxCR register.

Upon the trigger event, the corresponding generator channel starts generating DMA requests on its output. Each time the DMAMUX generated request is served by the connected DMA controller (a served request is deasserted), a built-in (inside the DMAMUX request generator) DMA request counter is decremented. At its underrun, the request generator channel stops generating DMA requests and the DMA request counter is automatically reloaded to its programmed value upon the next trigger event.

Thus, the number of DMA requests generated after the trigger event is GNBREQ + 1.

Note: The GNBREQ field value must be written by software only when the enable GE bit of the corresponding generator channel x is disabled.

A trigger event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_RGxCR register, the trigger events are masked during three AHB clock cycles.

Trigger overrun and interrupt

If a new DMA request trigger event occurs before the DMAMUX request generator counter underrun (the internal counter programmed via the GNBREQ field of the DMAMUX_RGxCR register), and if the request generator channel x was enabled via GE, then the request trigger event overrun flag bit OFx is asserted by the hardware in the status DMAMUX_RGSR register.

Note: The request generator channel x must be disabled (DMAMUX_RGxCR.GE = 0) at the completion of the usage of the related channel of the DMA controller. Else, upon a new detected trigger event, there is a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.

The overrun flag OFx is reset by setting the associated clear overrun flag bit COFx in the DMAMUX_RGCFR register.

Setting the DMAMUX request trigger overrun flag generates an interrupt if the DMA request trigger event overrun interrupt enable bit OIE is set in the DMAMUX_RGxCR register.

24.4 DMAMUX interrupts

An interrupt can be generated upon:

- A synchronization event overrun in each DMA request line multiplexer channel
- A trigger event overrun in each DMA request generator channel

For each case, per-channel individual interrupt enable, status and clear flag register bits are available.

Table 301. DMAMUX interrupts

Interrupt signal	Interrupt event	Event flag	Clear bit	Enable bit
dmamuxovr_it	Synchronization event overrun on channel x of the DMAMUX request line multiplexer	SOFx	CSOFx	SOIE
	Trigger event overrun on channel x of the DMAMUX request generator	OFx	COFx	OIE

24.5 DMAMUX registers

Refer to the table containing register boundary addresses for the DMAMUX base address.

DMAMUX registers may be accessed per (8-bit) byte, (16-bit) half-word, or (32-bit) word. The address must be aligned with the data size.

24.5.1 DMAMUX memory map

Table 302. DMAMUX register memory map

Offset	Register name
0x000 + 0x04 * x (x = 0 to 15)	*DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)*
0x080	*DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR)*
0x084	*DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR)*
0x100 + 0x04 * x (x = 0 to 3)	*DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)*
0x140	*DMAMUX request generator interrupt status register (DMAMUX_RGSR)*
0x144	*DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)*

24.5.2 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)

Address offset: 0x000 + 0x04 * x (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL[1:0]		SE
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	DMAREQ_ID[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SYNC_ID[4:0]**: Synchronization identification

Selects the synchronization input (see *Table 19: DMAMUX: assignment of multiplexer inputs to resources*).

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward

Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.

This field must only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity
Defines the edge polarity of the selected synchronization input:
00: No event, which means no synchronization nor detection.
01: Rising edge
10: Falling edge
11: Rising and falling edge

Bit 16 **SE**: Synchronization enable
0: Synchronization disabled
1: Synchronization enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EGE**: Event generation enable
0: Event generation disabled
1: Event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bits 7:0 **DMAREQ_ID[7:0]**: DMA request identification
Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

24.5.3 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SOF15	SOF14	SOF13	SOF12	SOF11	SOF10	SOF9	SOF8	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SOF[15:0]**: Synchronization overrun event flag
The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.
The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX_CFR register.

24.5.4 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSOF 15	CSOF 14	CSOF 13	CSOF 12	CSOF 11	CSOF 10	CSOF 9	CSOF 8	CSOF 7	CSOF 6	CSOF 5	CSOF 4	CSOF 3	CSOF 2	CSOF 1	CSOF 0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CSOF[15:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX_CSR register.

24.5.5 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)

Address offset: 0x100 + 0x04 * x (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL[1:0]		GE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	SIG_ID[4:0]				
							rw				rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ +1.

Note: This field must only be written when GE bit is disabled.

Bits 18:17 **GPOL[1:0]**: DMA request generator trigger polarity

Defines the edge polarity of the selected trigger input

00: No event which means none trigger detection nor generation.

01: Rising edge

10: Falling edge

11: Rising and falling edge

Bit 16 **GE**: DMA request generator channel x enable

0: DMA request generator channel x disabled

1: DMA request generator channel x enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **OIE**: Trigger overrun interrupt enable
0: Interrupt on a trigger overrun event occurrence is disabled
1: Interrupt on a trigger overrun event occurrence is enabled

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **SIG_ID[4:0]**: Signal identification
Selects the DMA request trigger input used for the channel x of the DMA request generator

24.5.6 DMAMUX request generator interrupt status register (DMAMUX_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF3	OF2	OF1	OF0
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **OF[3:0]**: Trigger overrun event flag
The flag is set when a new trigger event occurs on DMA request generator channel x, before the request counter underrun (the internal request counter programmed via the GNBREQ field of the DMAMUX_RGxCR register).
The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX_RGCFR register.

24.5.7 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF3	COF2	COF1	COF0
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **COF[3:0]**: Clear trigger overrun event flag
Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX_RGSR register.

25 Nested vectored interrupt controller (NVIC)

25.1 NVIC main features

- 168 maskable interrupt channels (not including the sixteen Cortex®-M7 interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

The NVIC and the processor core interface are closely coupled, which enable low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to the programming manual for Cortex®-M7 products.

25.2 SysTick calibration value register

The SysTick calibration value (SYST_CALIB) is fixed to 0x3E8. It provides a reference timebase of 1 ms based when the SysTick clock frequency is 1 MHz. To match the 1 ms timebase whatever the application frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- The SysTick clock source is the 100 MHz CPU clock (HCLK):
  reload_value = (F_{HCLK} x SYST_CALIB)-1
- or the SysTick clock source is an external clock:
  reload_value = ((F_{HCLK} /8) x SYST_CALIB)-1

where F_{HCLK} refers to the AHB frequency expressed in MHz.

For example, to achieve a timebase of 1 ms when the SysTick clock source is the 300 MHz HCLK:

reload_value = (300 x SYST_CALIB) = 0x493E0

26 Extended interrupt and event controller (EXTI)

The Extended Interrupt and event controller (EXTI) manages wakeup through configurable and direct event inputs. It generates interrupt requests to the CPU NVIC, and events to the CPU event input.

Both the interrupt request and event request generation can be used in Run modes.

26.1 EXTI main features

All Event inputs allow a CPU to wakeup and to generate a CPU interrupt and/or CPU event.

The asynchronous event inputs are classified in 2 groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse), they have the following features:
  - Selectable active trigger edge
  - Interrupt pending status register bit
  - Individual Interrupt and Event generation mask
  - SW trigger possibility
- Direct events (interrupt and wakeup sources from other peripherals, requiring to be cleared in the peripheral), they feature:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status is provided by the peripheral generating the event)
  - Individual Interrupt and Event generation mask
  - No SW trigger possibility

26.2 EXTI block diagram

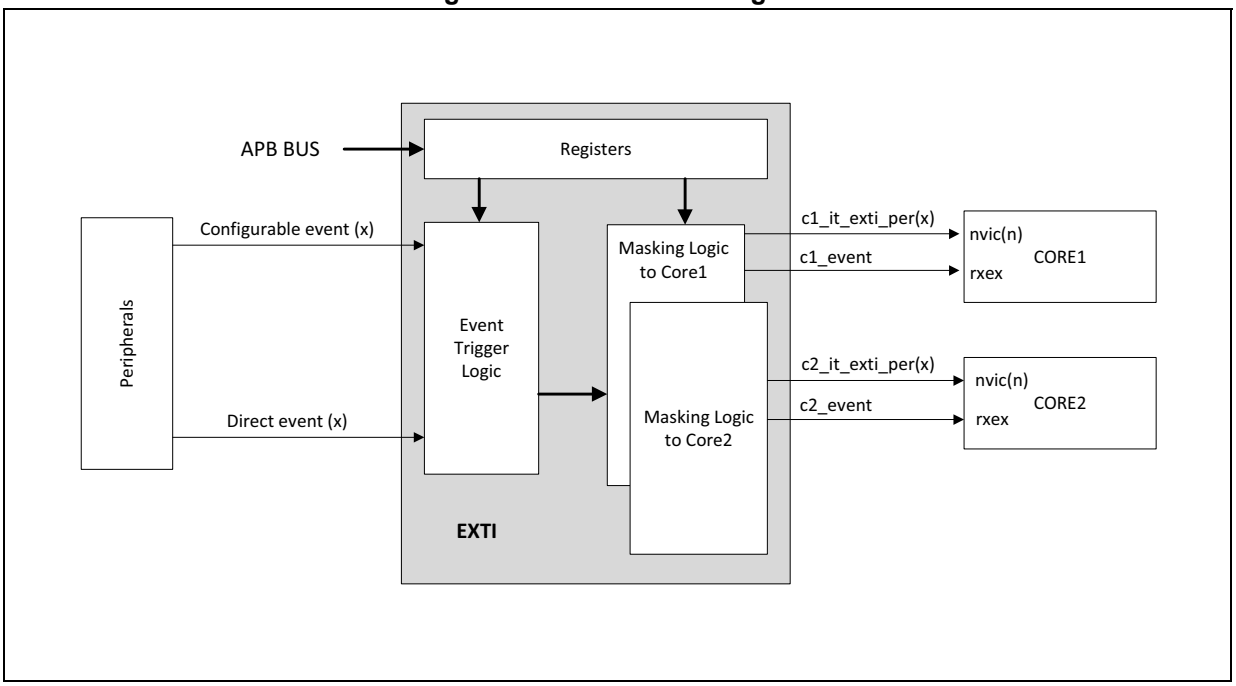
As shown in *Figure 226*, the EXTI consists of a register block accessed via an APB interface, an Event input Trigger block, and a Masking block.

The register block contains all EXTI registers.

The Event input trigger block provides Event input edge triggering logic.

The Masking block provides the Event input distribution to the different interrupt and event outputs, and their masking.

Figure 226. EXTI block diagram



26.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate events are connected to an EXTI Configurable event input or Direct Event input:

- Peripheral signals that generate a pulse are connected to an EXTI Configurable Event input. For these events the EXTI provides a CPU status pending bit that has to be cleared.
- Peripheral Interrupt sources that have to be cleared in the peripheral are connected to an EXTI Direct Event input. There is no CPU status pending bit within the EXTI. The Interrupt or Wakeup is cleared by the CPU in the peripheral.

The CPU(n) interrupts are connected to their respective CPU(n) NVIC, and, similarly, the CPU(n) event is connected to the CPU(n) rxev input.

26.3 EXTI functional description

Depending on the EXTI Event input type and wakeup target(s), different logic implementations are used. The applicable features are controlled from register bits:

- Active trigger edge enable, by *Rising trigger selection register (RTSR1)*, *Rising trigger selection register (RTSR2)*, and *Falling trigger selection register (FTSR1)*, *Falling trigger selection register (FTSR2)*
- Software trigger, by *Software interrupt event register (SWIER1)*, *Software interrupt event register (SWIER2)*
- CPU Interrupt enable, by *Interrupt mask register (CnIMR1)*, *Interrupt mask register (CnIMR2)*
- CPU Event enable, by *Event mask register (CnEMR1)*, *Event mask register (CnEMR2)*

Table 303. EXTI Event input configurations and register control

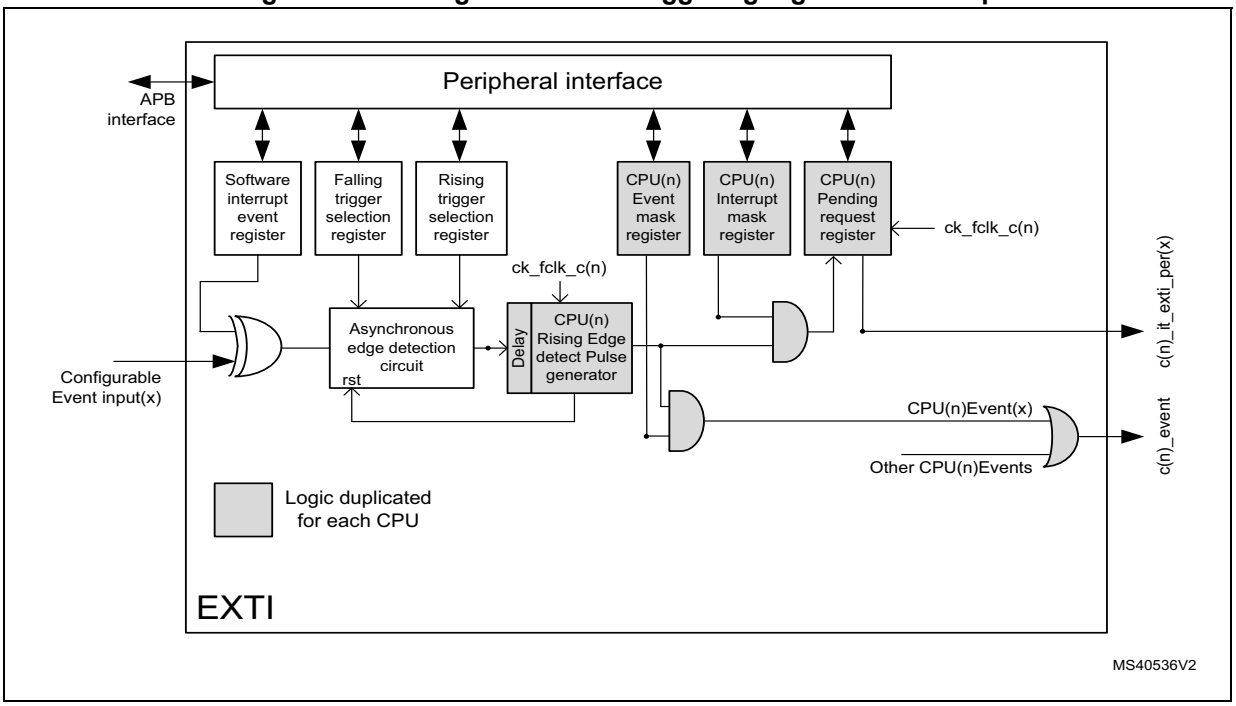
Event input type	Wakeup target(s)	Logic implementation	RTSR	FTSR	SWIER	CnIMR	CnEMR
Configurable	CPU(n)(1)	Configurable event input, CPU wakeup logic	X(2)	X	X	X	X
Direct	CPU(n)(1)	Direct event input, CPU wakeup logic	-	-	-	X	X

1. Waking-up CPU1 and/or CPU2.
2. X indicates that functionality is available.

26.3.1 EXTI configurable event input CPU wakeup

Figure 227 is a detailed representation of the logic associated with configurable Event inputs which always wake up a CPU(n).

Figure 227. Configurable event triggering logic CPU wakeup



The Software interrupt event register allows the system to trigger configurable events by software, writing the *Software interrupt event register (SWIER1)*, the *Software interrupt event register (SWIER2)* register bit.

The rising edge *Rising trigger selection register (RTSR1)*, *Rising trigger selection register (RTSR2)* and falling edge *Falling trigger selection register (FTSR1)*, *Falling trigger selection*

register (FTSR2) selection registers allow the system to enable and select the Configurable event active trigger edge or both edges.

Each CPU has its dedicated interrupt mask register, namely *Interrupt mask register (CnIMR1)* and *Interrupt mask register (CnIMR2)*, *Pending register (CnPR1)* and *Pending register (CnPR2)* for configurable events pending request registers. The CPU pending register is only set for an unmasked CPU(n) interrupt. Each event provides an individual CPU(n) interrupt to the CPU(n) NVIC. The configurable events interrupts need to be acknowledged by software in the CnPR register.

Each CPU has dedicated event mask registers, that is *Event mask register (CnEMR1)* and *Event mask register (CnEMR2)*. The enabled event then generates an event on a CPU. All events for a CPU are OR-ed together into a single CPU CPU(n) event signal. The CPU Pending register (CnPR) is not set for an unmasked CPU event.

When a CPU(n) interrupt or CPU(n) event is enabled, the Asynchronous edge detection circuit is reset by the clocked Delay and Rising edge detect pulse generator. This guarantees that the CPU(n) clock is woken up before the Asynchronous edge detection circuit is reset.

Note: A detected Configurable event, enabled by CPU(n), is only cleared when CPU(n) wakes up. When the CPU(n) is kept on hold (see Chapter 8: Power management), the detected configurable event is not cleared and the system is kept in Run mode. To clear the detected Configurable event the other CPU has to release the CPU(n) from hold.

26.3.2 EXTI configurable event input Any wakeup

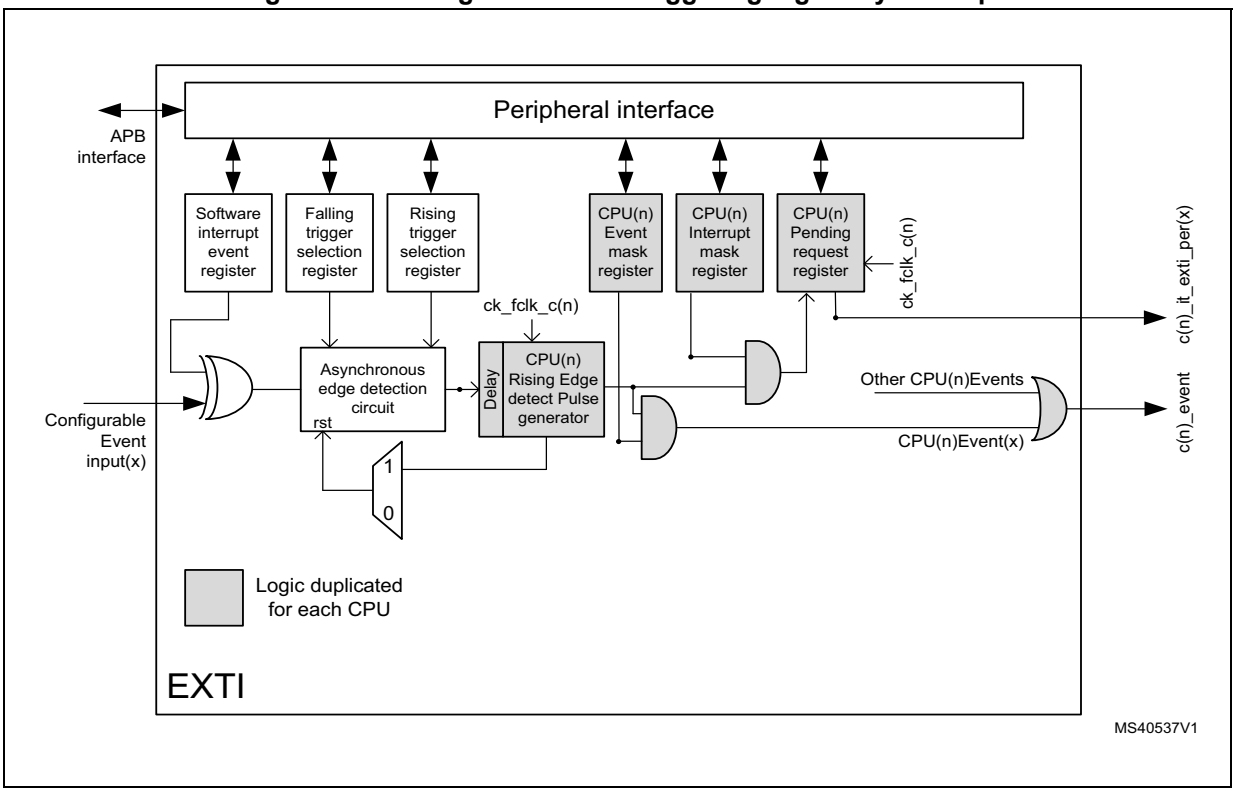
Figure 228 is a detailed representation of the logic associated with configurable event inputs that can wakeup CPU(n) (“Any” target). It provides the same functionality as the configurable event input CPU wakeup.

Table 304. Configurable Event input Asynchronous Edge detector reset

C1IMR	C1EMR	C2IMR	C2EMR	Asynchronous Edge detector reset by
Both = 0		Both = 0		Reserved
At least one = 1		Both = 0		CPU1 clock rising edge detect pulse generator
Both = 0		At least one = 1		CPU2 clock rising edge detect pulse generator
At least one = 1		At least one = 1		CPU1 clock rising edge detect pulse generator OR-ed(1) with CPU2 clock rising edge detect pulse generator

1. The first rising edge detect pulse generator resets the Asynchronous Edge detection circuit. A new configurable event input edge for both CPUs is only detected after the last rising edge detect pulse generator has completed. A configurable event input edge arriving between the detection of the first and the last rising edge detect pulse generator detection only signals a new event to the first CPU.

Figure 228. Configurable event triggering logic Any wakeup



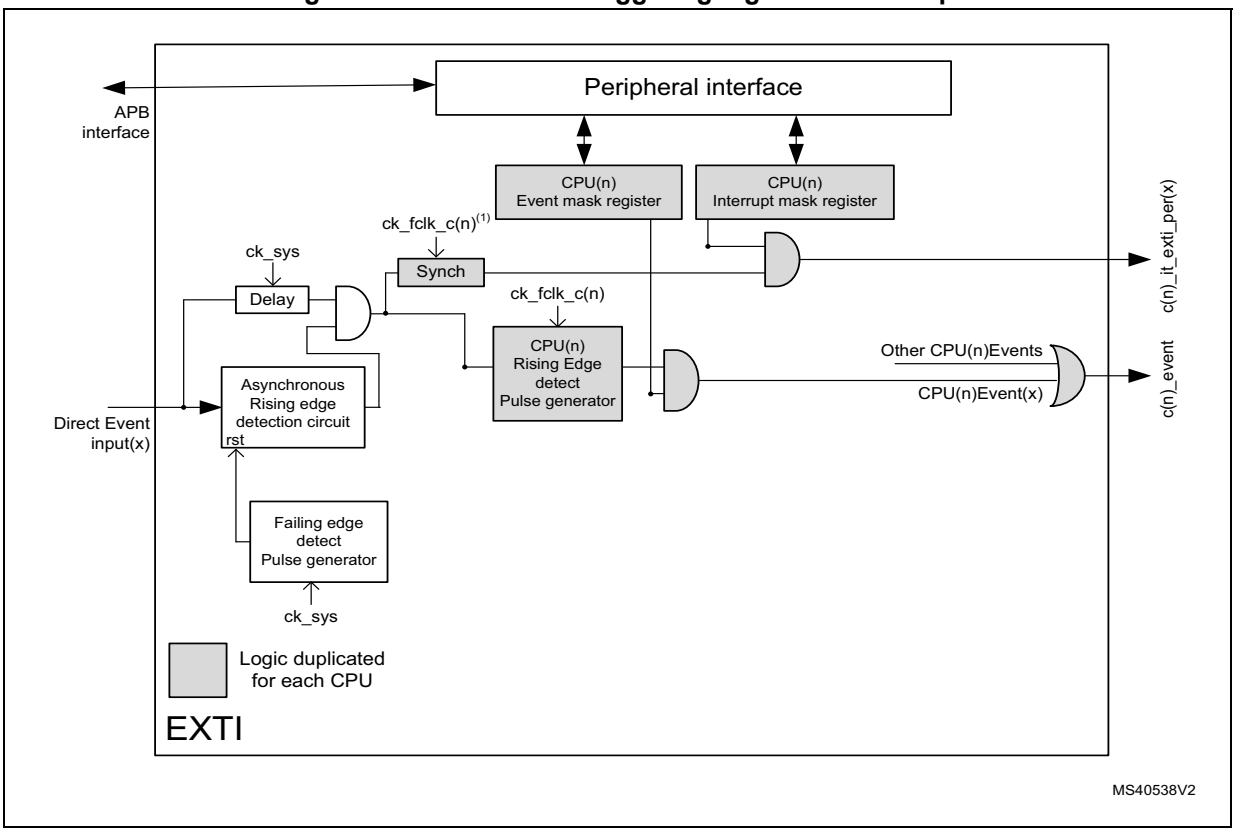
26.3.3 EXTI direct event input CPU wakeup

Figure 229 is a detailed representation of the logic associated with Direct Event inputs waking up a CPU(n).

Direct events only provide CPU(n) interrupt enable and CPU(n) event enable functionality.

Note: Direct events are cleared in the peripheral generating the event. When a Direct event input enabled by CPU(n) is cleared by the other CPU before the CPU(n) clock is running, (that is when CPU(n) is kept on hold, see Chapter 8: Power management), the CPU(n) no longer receives neither a CPU(n) interrupt nor a CPU(n) event, and does not wakeup. However the system stays in Run mode, generating the CPU(n) clock. For this reason CPU(n) Direct events must not be cleared by the other CPU.

Figure 229. Direct event triggering logic CPU Wakeup

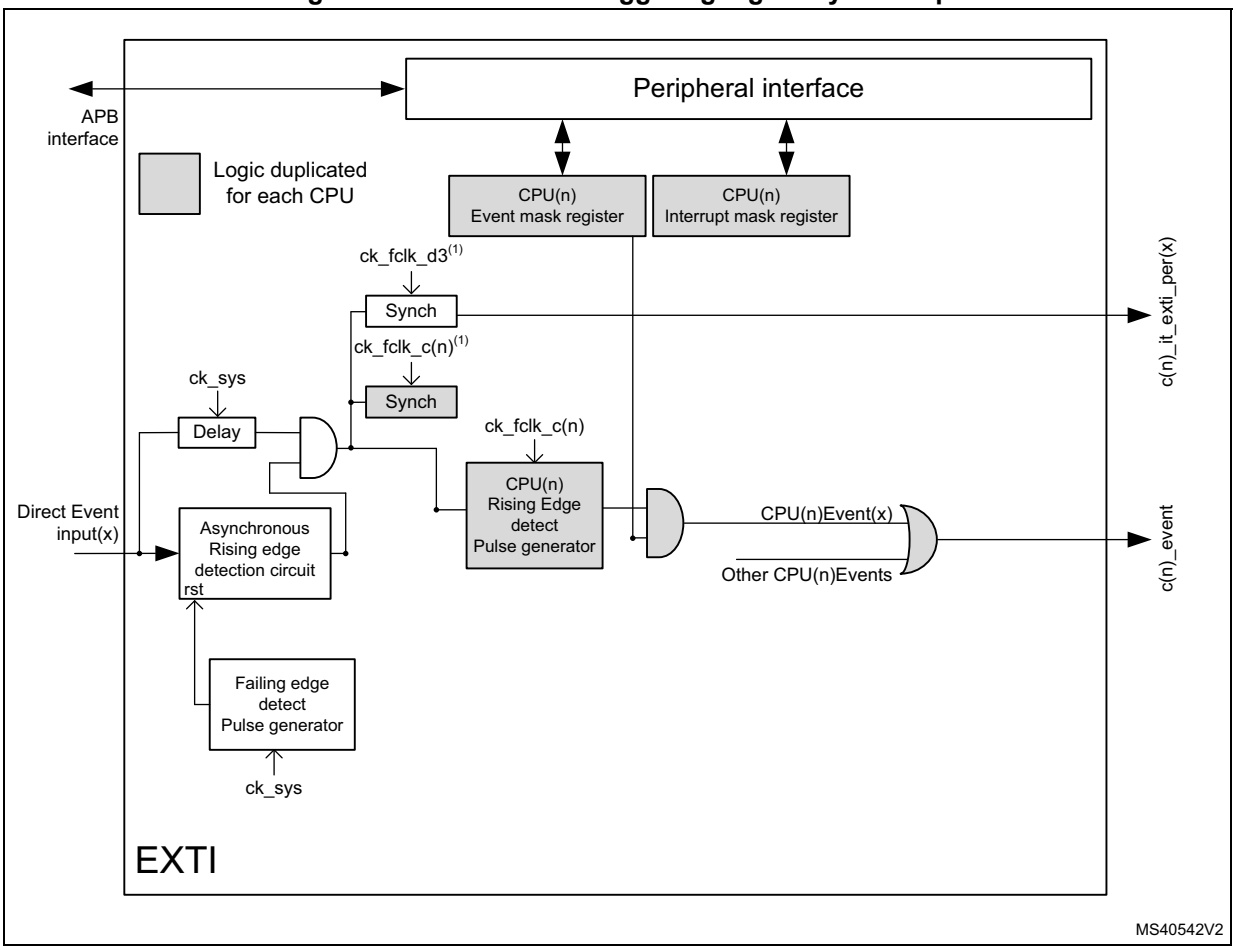


1. The CPU(n) interrupt for asynchronous Direct Event inputs (peripheral Wakeup signals) is synchronized with the CPU(n) clock. The synchronous Direct Event inputs (peripheral interrupt signals), after the asynchronous edge detection, are directly sent to the CPU(n) interrupt without resynchronization.

26.3.4 EXTI direct event input Any wakeup

Figure 230 is a detailed representation of the logic associated with Direct Event inputs waking up CPU(n), (“Any” target). It provides the same functionality as the Direct event input CPU wakeup.

Figure 230. Direct event triggering logic Any Wakeup



1. The CPU(n) interrupt for asynchronous Direct Event inputs (peripheral Wakeup signals) are synchronized, respectively, with the CPU(n) clock. The synchronous Direct Event inputs (peripheral interrupt signals), after the asynchronous edge detection, are directly sent to the CPU(n) interrupt without resynchronization in the EXTI.

26.4 EXTI event input mapping

For the sixteen GPIO Event inputs the associated IOPORT pin has to be selected in the SYSCFG register SYSCFG_EXTICRn. The same pin from each IOPORT maps to the corresponding EXTI Event input.

The wakeup capabilities of each Event input are detailed in *Table 17*. An Event input can either wake up CPU1, CPU2 or both.

The EXTI Event inputs with a connection to the CPU NVIC are indicated in the *Connection to NVIC* column. For the EXTI events not having a connection to the NVIC, the peripheral interrupt is directly connected to the NVIC in parallel with the connection to the EXTI.

All EXTI Event inputs are OR-ed together and connected to the CPU event input (rxev).

26.5 EXTI functional behavior

The Direct event inputs are enabled in the respective peripheral generating the event. The Configurable events are enabled by enabling at least one of the trigger edges.

An event only wakes up a CPU when the event associated CPU interrupt is unmasked and/or the CPU event is unmasked.

Table 305. Masking functionality

CPU(n)		Configurable event inputs PRx bits of CnPR	CPU(n)	
Interrupt enable MRx bits of CnIMR	Event enable MRx bits of CnEMR		Interrupt	Event
0	0	No	Masked	Masked
0	1	No	Masked	Yes
1	0	Status latched	Yes	Masked
1	1	Status latched	Yes	Yes

For configurable event inputs, when the enabled edge(s) occur on the event input, an event request is generated. When the associated CPU(n) interrupt is unmasked, the corresponding pending PRx bit in CnPR is set and the CPU(n) interrupt signal is activated. CnPR PRx pending bit must be cleared by software writing it to ‘1’. This clears the CPU(n) interrupt.

For Direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU(n) pending bit. When the associated CPU(n) interrupt is unmasked the corresponding CPU(n) interrupt signal is activated.

The CPU(n) event has to be unmasked to generate an event. When the enabled edge(s) occur on the Event input a CPU(n) event pulse is generated. There is no CPU(n) Event pending bit.

Both a CPU(n) interrupt and a CPU(n) event may be enabled on the same Event input. They both trigger the same Event input condition(s).

For the configurable event inputs, an event input request can be generated by software when writing a ‘1’ in the software interrupt/event register SWIER.

Whenever an Event input is enabled and a CPU(n) interrupt and/or CPU(n) event is unmasked.

26.5.1 EXTI CPU interrupt procedure

- Unmask the Event input interrupt by setting the corresponding mask bits in the CnIMR register.
- For configurable event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in RTSR and FTSR registers.
- Enable the associated interrupt source in the CPU(n) NVIC or use the SEVONPEND, so that an interrupt coming from the CPU(n) interrupt signal is detectable by the CPU after a WFI/WFE instruction.
  - For configurable event inputs the associated EXTI pending bit needs to be cleared.

26.5.2 EXTI CPU event procedure

- Unmask the Event input by setting the corresponding mask bits of the CnEMR register.
- For configurable event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in RTSR and FTSR registers.
- The CPU(n) event signal is detected by the CPU after a WFE instruction.
  - For configurable event inputs there is no EXTI pending bit to clear.

26.5.3 EXTI software interrupt/event trigger procedure

Any of the configurable event inputs can be triggered from the software interrupt/event register (the associated CPU(n) interrupt and/or CPU(n) event must be enabled by their respective procedure).

- Enable the Event input by setting at least one of the corresponding edge trigger bits in the RTSR and/or FTSR registers.
- Unmask the software interrupt/event trigger by setting at least one of the corresponding mask bits in the CnIMR and/or CnEMR registers.
- Trigger the software interrupt/event by writing “1” to the corresponding bit in the SWIER register.
- The Event input may be disabled by clearing the RTSR and FTSR register bits.

Note: An edge on the configurable event input also triggers an interrupt/event.

26.6 EXTI registers

Every register can only be accessed with 32-bit (word). A byte or half-word cannot be read or written.

26.6.1 EXTI memory map

Table 306. EXTI register memory map

Offset	Register name
0x00	*Rising trigger selection register (RTSR1)*
0x04	*Falling trigger selection register (FTSR1)*
0x08	*Software interrupt event register (SWIER1)*
0x20	*Rising trigger selection register (RTSR2)*
0x24	*Falling trigger selection register (FTSR2)*
0x28	*Software interrupt event register (SWIER2)*
0x80 (C1IMR1) 0xC0 (C2IMR1)	*Interrupt mask register (CnIMR1)*
0x84 (C1EMR1) 0xC4 (C2EMR1)	*Event mask register (CnEMR1)*
0x88 (C1PR1) 0xC8 (C2PR1)	*Pending register (CnPR1)*
0x90 (C1IMR2) 0xD0 (C2IMR2)	*Interrupt mask register (CnIMR2)*
0x94 (C1EMR2) 0xD4 (C2EMR2)	*Event mask register (CnEMR2)*
0x98 (C1PR2) 0xD8 (C2PR2)	*Pending register (CnPR2)*

26.6.2 Rising trigger selection register (RTSR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TR31	TR30	TR29	Res.	Res.	Res.	Res.	Res.	Res.	TR22	TR21	TR20	Res.	TR18	Res.	Res.
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TRx:** Rising trigger event configuration bit of Configurable Event input x.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:20 **TRx:** Rising trigger event configuration bit of Configurable Event input x.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TRx:** Rising trigger event configuration bit of Configurable Event input x.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:0 **TRx:** Rising trigger event configuration bit of Configurable Event input x.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

1. The Configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the Configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same Configurable Event input. In this case, both edges generate a trigger.

26.6.3 Falling trigger selection register (FTSR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TR31	TR30	TR29	Res.	Res.	Res.	Res.	Res.	Res.	TR22	TR21	TR20	Res.	TR18	Res.	Res.
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TRx:** Rising trigger event configuration bit of Configurable Event input x.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:20 **TRx:** Rising trigger event configuration bit of Configurable Event input x.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TRx:** Rising trigger event configuration bit of Configurable Event input x.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:0 **TRx:** Rising trigger event configuration bit of Configurable Event input x.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the Configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same Configurable Event input. In this case, both edges generate a trigger.

26.6.4 Software interrupt event register (SWIER1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWIER 31	SWIER 30	SWIER 29	Res.	Res.	Res.	Res.	Res.	Res.	SWIER 22	SWIER 21	SWIER 20	Res	SWIER 18	Res.	Res.
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **SWIERx:** Software interrupt on line x
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:20 **SWIERx:** Software interrupt on line x
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

Bits 19 Reserved, must be kept at reset value.

Bits 18 **SWIERx:** Software interrupt on line x
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:0 **SWIERx:** Software interrupt on line x
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

26.6.5 Rising trigger selection register (RTSR2)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR34	TR33	TR32
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 3:0 **TRx:** Rising trigger event configuration bit of Configurable Event input x+32.(1)

0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the Configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same Configurable Event input. In this case, both edges generate a trigger.

26.6.6 Falling trigger selection register (FTSR2)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR34	TR33	TR32
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 3:0 **TRx:** Rising trigger event configuration bit of Configurable Event input x+32.(1)

0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

26.6.7 Software interrupt event register (SWIER2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER 34	SWIER 33	SWIER 32
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 3:0 **SWIERx:** Software interrupt on line x+32
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

26.6.8 Interrupt mask register (CnIMR1)

Address offset: 0x80 (C1IMR1), 0xC0 (C2IMR1)

Reset value: 0x1780 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR31	MR30	MR29	MR28	Res	MR26	MR25	MR24	MR23	MR22	MR21	MR20	Res	MR18	Res	Res
rw	rw	rw							rw	rw	rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **MRx:** CPUn interrupt Mask on Configurable Event input x
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bits 28 **MRx:** CPUn interrupt mask on Direct Event input x(1)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bits 27 Reserved, must be kept at reset value.

Bits 26:23 **MRx:** CPUn interrupt mask on Direct Event input x
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bits 22:20 **MRx:** CPUn interrupt Mask on Configurable Event input x (2)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bit 19 Reserved, must be kept at reset value.

Bits 18 **MRx:** CPUn interrupt Mask on Configurable Event input x
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:0 **MRx:** CPUn interrupt Mask on Configurable Event input x
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

1. The reset value for Direct Event inputs is set to ‘1’ in order to enable the interrupt by default.
2. The reset value for Configurable Event inputs is set to ‘0’ in order to disable the interrupt by default.

26.6.9 Event mask register (CnEMR1)

Address offset: 0x84 (C1EMR1), 0xC4 (C2EMR1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR31	MR30	MR29	MR28	Res.	MR26	MR25	MR24	MR23	MR22	MR21	MR20	Res.	MR18	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **MRx:** CPUn Event mask on Event input x
0: Event request from Line x is masked
1: Event request from Line x is unmasked

Bits 28 **MRx:** CPUn interrupt mask on Direct Event input x
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bits 27 Reserved, must be kept at reset value.

Bits 26:23 **MRx:** CPUn interrupt mask on Direct Event input x
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bits 22:20 **MRx:** CPUn Event mask on Event input x
0: Event request from Line x is masked
1: Event request from Line x is unmasked

Bit 19 Reserved, must be kept at reset value.

Bits 18 **MRx:** CPUn Event mask on Event input x
0: Event request from Line x is masked
1: Event request from Line x is unmasked

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:0 **MRx:** CPUn Event mask on Event input x
0: Event request from Line x is masked
1: Event request from Line x is unmasked

26.6.10 Pending register (CnPR1)

Address offset: 0x88 (C1PR1), 0xC8 (C2PR1)

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PR31	PR30	PR29	Res.	Res.	Res.	Res.	Res.	Res.	PR22	PR21	PR20	Res.	PR18	Res.	Res.
rc1	rc1	rc1							rc1	rc1	rc1		rc1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1

Bits 31:29 **PRx:** Configurable event inputs x Pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:20 **PRx:** Configurable event inputs x Pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 19 Reserved, must be kept at reset value.

Bits 18 **PRx:** Configurable event inputs x Pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:0 **PRx:** Configurable event inputs x Pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

26.6.11 Interrupt mask register (CnIMR2)

Address offset: 0x90 (C1IMR2), 0xD0 (C2IMR2)

Reset value: 0x0000 3000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MR45	MR44	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR34	MR33	MR32
													rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13:12 **MR45:44:** CPUn interrupt Mask on Direct Event input 44:45(1)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bits 11:3 Reserved, must be kept at reset value.

Bit 2:0 **MRx:** CPUn interrupt Mask on Configurable Event input x+32(2)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

1. The reset value for Direct Event inputs is set to '1' in order to enable the interrupt by default.
2. The reset value for Configurable Event inputs is set to ‘0’ in order to disable the interrupt by default.

26.6.12 Event mask register (CnEMR2)

Address offset: 0x94 (C1EMR2), 0xD4 (C2EMR2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MR45	MR44	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR34	MR33	MR32
													rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13:12 **MR45:44:** CPUn interrupt Mask on Direct Event input 44:45[1]

0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bits 11:3 Reserved, must be kept at reset value.

Bit 2:0 **MRx:** CPUn Event mask on Event input x+32

0: Event request from Line x is masked
1: Event request from Line x is unmasked

1. The reset value for Direct Event inputs is set to '1' in order to enable the interrupt by default.

26.6.13 Pending register (CnPR2)

Address offset: 0x98 (C1PR2), 0xD8 (C2PR2)

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR34	PR33	PR32
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2:0 **PRx:** Configurable event inputs x+32 Pending bit

0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

27 Cyclic redundancy check calculation unit (CRC)

27.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

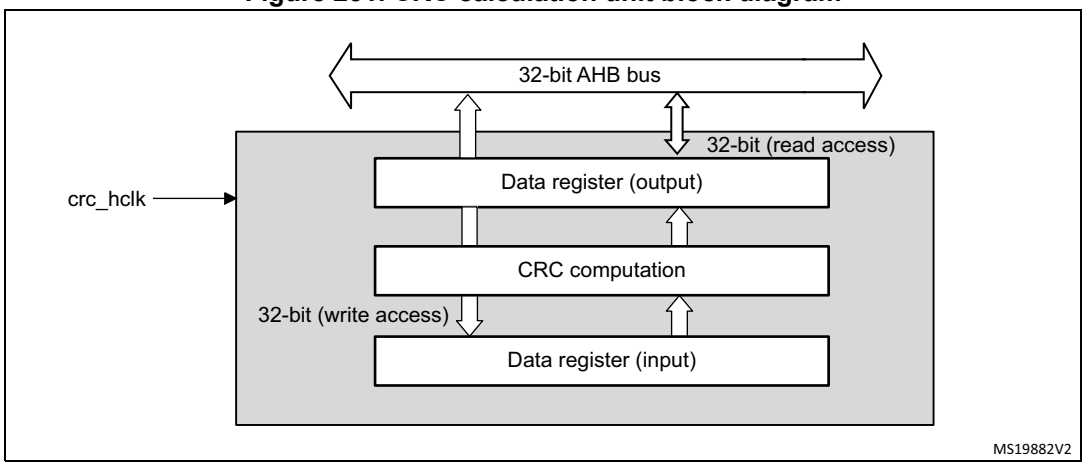
27.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data

27.3 CRC functional description

27.3.1 CRC block diagram

Figure 231. CRC calculation unit block diagram



27.3.2 CRC internal signals

Table 307. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

27.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit access is allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32-bit
- 2 AHB clock cycles for 16-bit
- 1 AHB clock cycles for 8-bit

An input buffer allows a second data to be immediately written without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example: input data 0x1A2B3C4D is used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level: for example, output data 0x11223344 is converted into 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size cannot be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

27.4 CRC registers

27.4.1 CRC memory map

Table 308. CRC register memory map

Offset	Register name
0x00	*CRC data register (CRC_DR)*
0x04	*CRC independent data register (CRC_IDR)*
0x08	*CRC control register (CRC_CR)*
0x10	*CRC initial value (CRC_INIT)*
0x14	*CRC polynomial (CRC_POL)*

27.4.2 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.
It holds the previous CRC calculation result when it is read.
If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

27.4.3 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits
These bits can be used as a temporary storage location for four bytes.
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

27.4.4 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rs

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **REV_OUT**: Reverse output data
This bit controls the reversal of the bit order of the output data.
0: Bit order not affected
1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data
These bits control the reversal of the bit order of the input data
00: Bit order not affected
01: Bit reversal done by byte
10: Bit reversal done by half-word
11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size
These bits control the size of the polynomial.
00: 32 bit polynomial
01: 16 bit polynomial
10: 8 bit polynomial
11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit
This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

27.4.5 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_INIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRC_INIT[31:0]***: Programmable initial CRC value*

This register is used to write the CRC initial value.

27.4.6 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **POL[31:0]**: Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

28 CORDIC coprocessor (CORDIC)

28.1 CORDIC introduction

The CORDIC coprocessor provides hardware acceleration of certain mathematical functions (mainly trigonometric ones) commonly used in motor control, metering, signal processing and many other applications.

It speeds up the calculation of these functions compared to a software implementation, making it possible the use of a lower operating frequency, or freeing up processor cycles in order to perform other tasks.

28.2 CORDIC main features

- 24-bit CORDIC rotation engine
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Functions: sine, cosine, sinh, cosh, atan, atan2, atanh, modulus, square root, natural logarithm
- Programmable precision
- Low latency AHB slave interface
- Results can be read as soon as ready, without polling or interrupt
- DMA read and write channels
- Multiple register read/write by DMA

28.3 CORDIC functional description

28.3.1 General description

The CORDIC is a cost-efficient successive approximation algorithm for evaluating trigonometric and hyperbolic functions.

In trigonometric (circular) mode, the sine and cosine of an angle θ are determined by rotating the unit vector [1, 0] through decreasing angles until the cumulative sum of the rotation angles equals the input angle θ. The x and y Cartesian components of the rotated vector then correspond respectively to the cosine and sine of θ. Inversely, the angle of a vector [x, y], corresponding to arctangent (y / x), is determined by rotating [x, y] through successively decreasing angles to obtain the unit vector [1, 0]. The cumulative sum of the rotation angles gives the angle of the original vector.

The CORDIC algorithm can also be used for calculating hyperbolic functions (sinh, cosh, atanh), by replacing the successive circular rotations by steps along a hyperbole.

Other functions can be derived from the basic functions described above.

28.3.2 CORDIC functions

The first step when using the coprocessor is to select the required function, by programming the FUNC field of the CORDIC_CR register accordingly.

The table below lists the functions supported by the CORDIC coprocessor.

Table 309. CORDIC functions

Function	Primary argument (ARG1)	Secondary argument (ARG2)	Primary result (RES1)	Secondary result (RES2)
Cosine	angle θ	modulus m	$m \cdot \cos\theta$	$m \cdot \sin\theta$
Sine	angle θ	modulus m	$m \cdot \sin\theta$	$m \cdot \cos\theta$
Phase	x	y	atan2(y,x)	$\sqrt{x^2+y^2}$
Modulus	x	y	$\sqrt{x^2+y^2}$	atan2(y,x)
Arctangent	x	none	$\tan^{-1} x$	none
Hyperbolic cosine	x	none	cosh x	sinh x
Hyperbolic sine	x	none	sinh x	cosh x
Hyperbolic arctangent	x	none	$\tanh^{-1} x$	none
Natural logarithm	x	none	ln x	none
Square root	x	none	$\sqrt{x}$	none

Several functions take two input arguments, ARG1 and ARG2, and some generate two results simultaneously, RES1 and RES2. This is a side-effect of the algorithm and means that only one operation is needed to obtain two values. This is the case, for example, when performing polar-to-rectangular conversion: sin θ also generates cos θ, while cos θ also generates sin θ. Similarly for rectangular-to-polar conversion (phase(x,y), modulus(x,y)) and for hyperbolic functions (cosh θ, sinh θ).

Note: The exponential function, exp x, can be obtained as the sum of sinh x and cosh x. Furthermore, base N logarithms, $\log_N x$, can be derived by multiplying ln x by a constant K, where K = 1/ln N.

For certain functions (atan, log, sqrt) a scaling factor (see *Section 28.3.4*) can be applied to extend the range of the function beyond the maximum [-1, 1] supported by the q1.31 fixed point format. The scaling factor must be set to 0 for all other circular functions, and to 1 for hyperbolic functions.

Cosine

Table 310. Cosine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m · cos θ	[-1, 1]

Table 310. Cosine parameters (continued)

Parameter	Description	Range
RES2	m · sin θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the cosine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument m is the modulus, m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the sine of the angle, multiplied by the modulus.

Sine

Table 311. Sine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m · sin θ	[-1, 1]
RES2	m · cos θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the sine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument m is the modulus, m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the sine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.

Phase

Table 312. Phase parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Phase angle θ in radians, divided by π	[-1, 1]

Table 312. Phase parameters (continued)

Parameter	Description	Range
RES2	Modulus m	[0, 1]
SCALE	Not applicable	0

This function calculates the phase angle in the range -π to π of a vector **v** = [x y] (also known as atan2(y,x). It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the phase angle θ of the vector **v**. RES1 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

The secondary result, RES2, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES2 is saturated to 1.

Modulus

Table 313. Modulus parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Modulus m	[0, 1]
RES2	Phase angle θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the magnitude, or modulus, of a vector **v** = [x y]. It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES1 is saturated to 1.

The secondary result, RES2, is the phase angle θ of the vector **v**. RES2 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

Arctangent

Table 314. Arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-1, 1]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \tan^{-1} x$, in radians, divided by p	[-1, 1]
RES2	Not applicable	-
SCALE	n	[0 7]

This function calculates the arctangent, or inverse tangent, of the input argument x.

The primary argument, ARG1, is the input value, $x = \tan \theta$. If $|x| > 1$, a scaling factor of 2^{-n} must be applied in software such that $-1 < x \cdot 2^{-n} < 1$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the scale factor n must be programmed in the SCALE parameter.

Note that the maximum input value allowed is $\tan \theta = 128$, which corresponds to an angle $\theta = 89.55$ degrees. For $|x| > 128$, a software method must be used to find $\tan^{-1} x$.

The secondary argument, ARG2, is unused.

The primary result, RES1, is the angle $\theta = \tan^{-1} x$. RES1 must be multiplied by $2^n \cdot \pi$ to obtain the angle in radians.

The secondary result, RES2, is unused.

Hyperbolic cosine

Table 315. Hyperbolic cosine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \cosh x$	[0.5 0.846]
RES2	$2^{-n} \cdot \sinh x$	[-0.683 0.683]
SCALE	n	1

This function calculates the hyperbolic cosine of a hyperbolic angle x. It can also be used to calculate the exponential functions $e^x = \cosh x + \sinh x$, and $e^{-x} = \cosh x - \sinh x$.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. Since the minimum value of cosh x is 1, which is beyond the range of the q1.31 format, a scaling factor of 2^{-n} must be applied in software. The factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic cosine, cosh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic sine, sinh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic sine

Table 316. Hyperbolic sine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559, 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \sinh x$	[-0.683, 0.683]
RES2	$2^{-n} \cdot \cosh x$	[0.5, 0.846]
SCALE	n	1

This function calculates the hyperbolic sine of a hyperbolic angle x. It can also be used to calculate the exponential functions e^x = cosh x + sinh x, and e^{-x} = cosh x- sinh x.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. For all input values, a scaling factor of 2^{-n} must be applied in software, where n = 1. The scaled value x · 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic sine, sinh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic cosine, cosh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic arctangent

Table 317. Hyperbolic arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.403 0.403]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \text{atanh } x$	[-0.559 0.559]
RES2	Not applicable	-
SCALE	n	1

This function calculates the hyperbolic arctangent of the input argument x.

The primary argument is the input value x. Only values of x in the range -0.806 to +0.806 are supported. The value x must be scaled by a factor 2^{-n}, where n = 1. The scaled value

$x \cdot 0.5$ is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result is the hyperbolic arctangent, atanh x. RES1 must be multiplied by 2 to obtain the correct value.

The secondary result is not used.

Natural logarithm

Table 318. Natural logarithm parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.054 0.875]
ARG2	Not applicable	-
RES1	$2^{-(n+1)}.\ln x$	[-0.279 0.137]
RES2	Not applicable	-
SCALE	n	[1 4]

This function calculates the natural logarithm of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.107 to 9.35 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < 1-2^{-n}$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 319 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 319. Natural log scaling factors and corresponding ranges

n	x range	ARG1 range
1	$0.107 \leq x < 1$	$0.0535 \leq ARG1 < 0.5$
2	$1 \leq x < 3$	$0.25 \leq ARG1 < 0.75$
3	$3 \leq x < 7$	$0.375 \leq ARG1 < 0.875$
4	$7 \leq x \leq 9.35$	$0.4375 \leq ARG1 < 0.584$

The secondary argument is not used.

The primary result is the natural logarithm, ln x. RES1 must be multiplied by $2^{(n+1)}$ to obtain the correct value.

The secondary result is not used.

Square root

Table 320. Square root parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.027 0.875]
ARG2	Not applicable	-
RES1	$2^{-n}\sqrt{x}$	[0.04 1]
RES2	Not applicable	-
SCALE	n	[0 2]

This function calculates the square root of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.027 to 2.34 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < (1 - 2^{(-n-2)})$.

The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 321 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 321. Square root scaling factors and corresponding ranges

n	x range	ARG1 range
0	$0.027 \leq x < 0.75$	0.027 ≤ ARG1 < 0.75
1	$0.75 \leq x < 1.75$	0.375 ≤ ARG1 < 0.875
2	$1.75 \leq x \leq 2.341$	0.4375 ≤ ARG1 ≤ 0.585

The secondary argument is not used.

The primary result is the square root of x. RES1 must be multiplied by 2^n to obtain the correct value.

The secondary result is not used.

28.3.3 Fixed point representation

The CORDIC operates in fixed point signed integer format. Input and output values can be either q1.31 or q1.15.

In q1.31 format, numbers are represented by one sign bit and 31 fractional bits (binary decimal places). The numeric range is therefore -1 (0x80000000) to $1 - 2^{-31}$ (0x7FFFFFFF).

In q1.15 format, the numeric range is 1 (0x8000) to $1 - 2^{-15}$ (0x7FFF). This format has the advantage that two input arguments can be packed into a single 32-bit write, and two results can be fetched in one 32-bit read.

28.3.4 Scaling factor

Several of the functions listed in *Section 28.3.2* specify a scaling factor, SCALE. This allows the function input range to be extended to cover the full range of values supported by the CORDIC, without saturating the input, output or internal registers. If the scaling factor is

required, it has to be calculated in software and programmed into the SCALE field of the CORDIC_CSR register. The input arguments must be scaled accordingly before programming the scaled values in the CORDIC_WDATA register. The scaling must also be undone on the results read from the CORDIC_RDATA register.

Note: The scaling factor entails a loss of precision due to truncation of the scaled value.

28.3.5 Precision

The precision of the result is dependent on the number of CORDIC iterations. The algorithm converges at a constant rate of one binary digit per iteration for trigonometric functions (sine, cosine, phase, modulus), see *Figure 232*.

For hyperbolic functions (hyperbolic sine, hyperbolic cosine, natural logarithm), the convergence rate is less constant due to the peculiarities of the CORDIC algorithm (see *Figure 233*). The square root function converges at roughly twice the speed of the hyperbolic functions (see *Figure 234*).

Figure 232. CORDIC convergence for trigonometric functions

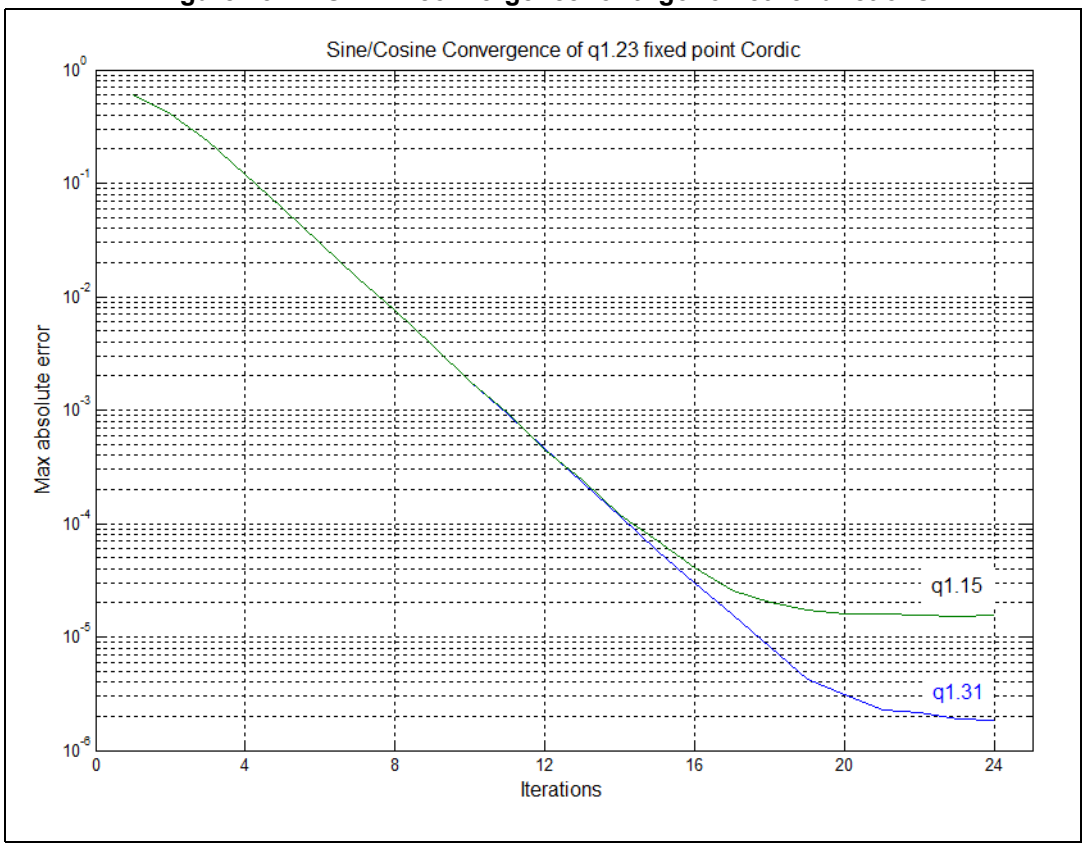


Figure 233. CORDIC convergence for hyperbolic functions

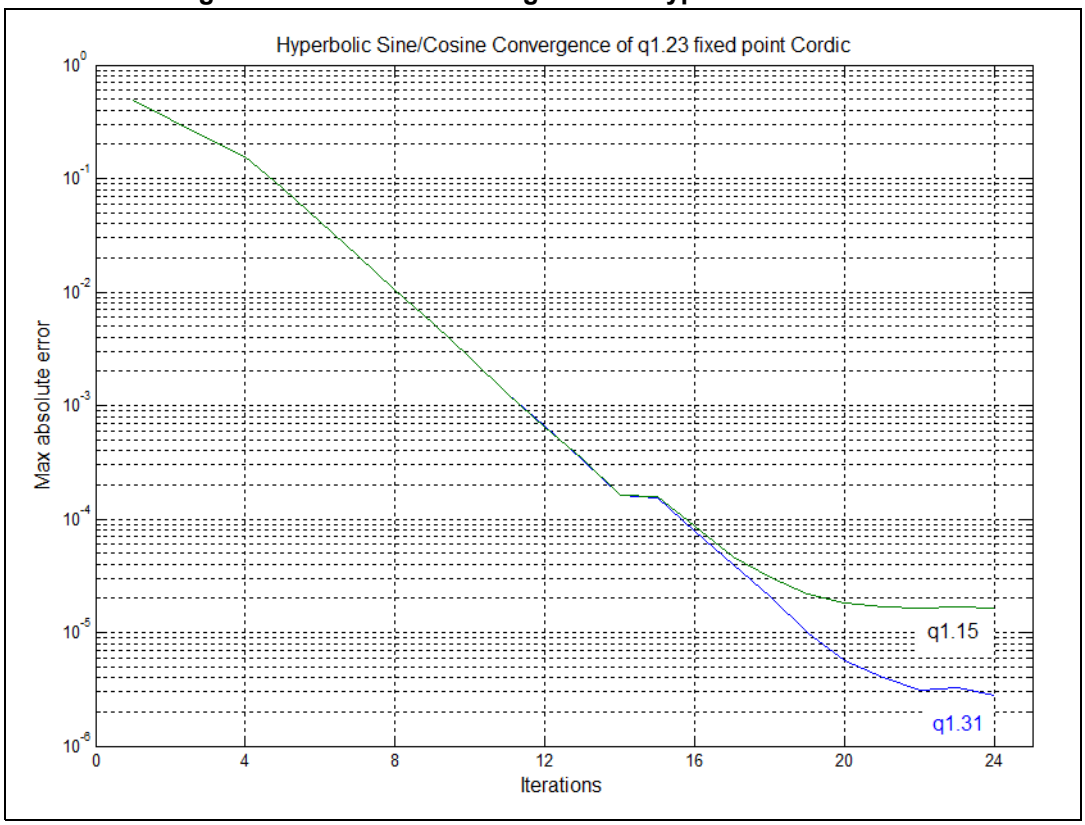
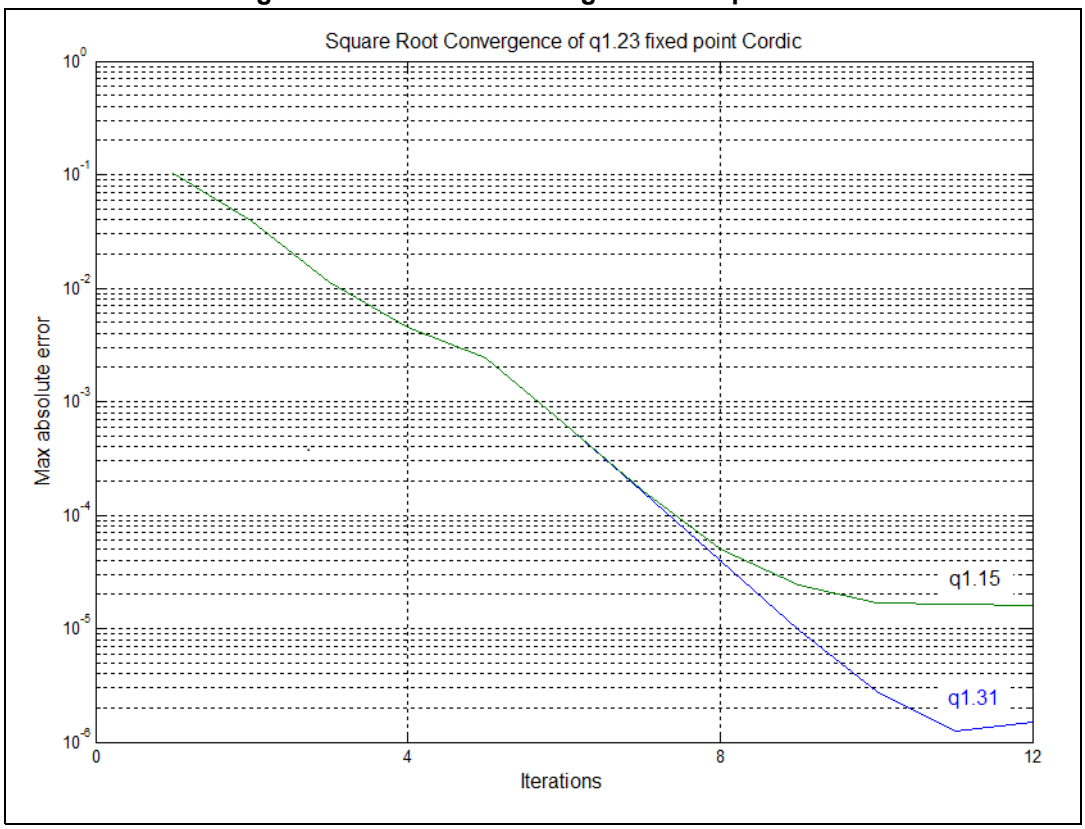


Figure 234. CORDIC convergence for square root



Note: The convergence rate decreases as the quantization error starts to become significant.

The CORDIC can perform four iterations per clock cycle. For each function, the maximum error remaining after every four iterations is shown in *Table 322*, together with the number of clock cycles required to reach that precision. From this table, the desired number of cycles can be determined and programmed in the PRECISION field of the CORDIC_CR register. The coprocessor stops as soon as the programmed number of iterations has completed, and the result can be read immediately.

Table 322. Precision vs. number of iterations

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sin, Cos, Phase(2), Mod, Atan(4)	4	1	2^{-3}	2^{-3}
	8	2	2^{-7}	2^{-7}
	12	3	2^{-11}	2^{-11}
	16	4	2^{-15}	2^{-15}
	20	5	2^{-18}	2^{-16}
	24	6	2^{-19}	2^{-16}

Table 322. Precision vs. number of iterations (continued)

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sinh, Cosh, Atanh, Ln(3)	4	1	2^{-2}	2^{-2}
	8	2	2^{-6}	2^{-6}
	12	3	2^{-10}	2^{-10}
	16	4	2^{-13}	2^{-13}
	20	5	2^{-17}	2^{-15}
	24	6	2^{-18}	2^{-15}
Sqrt(4)	4	1	2^{-7}	2^{-7}
	8	2	2^{-14}	2^{-14}
	12	3	2^{-19}	2^{-15}

1. Max residual error is the maximum error remaining after the given number of iterations, compared to the identical calculation performed in double precision floating point. An additional rounding error may be incurred, of up to 2^{-16} for q15 format or 2^{-20} for q31 format.
2. For modulus > 0.5. The achievable precision reduces proportionally to the magnitude of the modulus, as quantization error becomes significant.
3. SCALE = 1. If a higher scaling factor is used, the achievable precision is reduced proportionally.
4. SCALE = 0. If a higher scaling factor is used, the achievable precision is reduced proportionally.

28.3.6 Zero-overhead mode

The fastest way to use the coprocessor is to pre-program the CORDIC_CSR register with the function to be performed (FUNC), the desired number of clock cycles (PRECISION), the size of the input and output values (ARGSIZE, RESSIZE), the number of input arguments (NARGS) and/or results (NRES), and the scaling factor (SCALE), if applicable.

Subsequently, a calculation is triggered by writing the input arguments to the CORDIC_WDATA register. As soon as the correct number of input arguments has been written (and any ongoing calculation has finished) a new calculation is launched using these input arguments and the current CORDIC_CSR settings. There is no need to re-program the CORDIC_CSR register if there is no change.

If a dual 32-bit input argument is needed (ARGSIZE = 0, NARGS = 1), the primary input argument, ARG1, must be written first, followed by the secondary argument, ARG2. If the secondary argument remains unchanged for a series of calculations, the second write can be avoided, by reprogramming the number of arguments to one (NARGS = 0), once the first calculation has started. The secondary argument retains its programmed value as long as the function is not changed.

Note: ARG2 is set to +1 (0x7FFFFFFF) after a reset.

If two 16-bit arguments are used (ARGSIZE = 1) they must be packed into a 32-bit word, with ARG1 in the least significant half-word and ARG2 in the most significant half-word. The packed 32-bit word is then written to the CORDIC_WDATA register. Only one write is needed in this case (NARGS = 0).

For functions which take only one input argument, ARG1, it is recommended to set NARGS = 0. If NARGS = 1, a second write to CORDIC_WDATA must be performed to trigger the calculation. The ARG2 data in this case is not used.

Once the calculation has started, any attempt to read the CORDIC_RDATA register inserts bus wait states until the calculation has finished, before returning the result. Hence it is possible for the software to write the input and immediately read the result without polling to see if it is valid. Alternatively, the processor can wait for the appropriate number of clock cycles before reading the result. This time can be used to program the CORDIC_CSR register for the next calculation and prepare the next input data, if needed. The CORDIC_CSR register can be re-programmed while a calculation is in progress, without affecting the result of the ongoing calculation. In the same way, the CORDIC_WDATA register can be updated with the next argument(s) once the previous arguments have been taken into account. The next arguments and settings remain pending until the previous calculation has completed.

When a calculation is finished, the result(s) can be read from the CORDIC_RDATA register. If two 32-bit results are expected (NRES = 1, RESSIZE = 0), the primary result (RES1) is read out first, followed by the secondary result (RES2). If only one 32-bit result is expected (NRES = 0, RESSIZE = 0), then RES1 is output on the first read.

If 16-bit results are expected (RESSIZE = 1), a single read to CORDIC_RDATA fetches both results packed into a 32-bit word. RES1 is in the lower half-word, and RES2 in the upper half-word. In this case, it is recommended to program NRES = 0. IF NRES = 1, a second read of CORDIC_RDATA must be performed in order to free up the CORDIC for the next operation. The data from this second read must be discarded.

The next calculation starts when the expected number of results has been read, provided the expected number of arguments have been written. This means that at any time, there can be one calculation in progress, or waiting for the results to be read, and one operation pending. Any further access to CORDIC_WDATA while an operation is pending, cancels the pending operation and overwrite the data.

The following sequence summarizes the use of the CORDIC_IP in zero-overhead mode:

1. Program the CORDIC_CSR register with the appropriate settings
2. Program the argument(s) for the first calculation in the CORDIC_WDATA register. This launches the first calculation.
3. If needed, update the CORDIC_CSR register settings for the next calculation.
4. Program the argument(s) for the next calculation in the CORDIC_WDATA register.
5. Read the result(s) from the CORDIC_RDATA register. This triggers the next calculation.
6. Go to step *3*.

28.3.7 Polling mode

When a new result is available in the CORDIC_RDATA register, the RRDY flag is set in the CORDIC_CSR register. The flag can be polled by reading the register. It is reset by reading the CORDIC_RDATA register (once or twice depending on the NRES field of the CORDIC_CSR register).

Polling the RRDY flag takes slightly longer than reading the CORDIC_RDATA register directly, since the result is not read as soon as it is available. However the processor and bus interface are not stalled while reading the CORDIC_CSR register, so this mode may be of interest if stalling the processor is not acceptable (for example if low latency interrupts must be serviced).

28.3.8 Interrupt mode

By setting the interrupt enable (IE) bit in the CORDIC_CSR register, an interrupt is generated whenever the RRDY flag is set. The interrupt is cleared when the flag is reset.

This mode allows the result of the calculation to be read under interrupt service routine, and hence given a priority relative to other tasks. However it is slower than directly reading the result, or polling the flag, due to the interrupt handling delays.

28.3.9 DMA mode

If the DMA write enable (DMAWEN) bit is set in the CORDIC_CSR register, and no operation is pending, a DMA write channel request is generated. The DMA controller can transfer a primary input argument (ARG1) from memory into the CORDIC_WDATA register. Writing into the register deasserts the DMA request. If NARGS = 1 in the CORDIC_CSR register, a second DMA write channel request is generated to transfer the secondary input argument (ARG2) into the CORDIC_WDATA register. When all input arguments have been written, and any ongoing calculation has been completed (by reading the results), a new calculation is started and another DMA write channel request is generated.

If the DMA read enable (DMAREN) bit is set in the CORDIC_CSR register, the RRDY flag going active generates a DMA read channel request. The DMA controller can then transfer the primary result (RES1) from the CORDIC_RDATA register to memory. Reading the register deasserts the DMA request. If NRES = 1 in the CORDIC_CSR register, a second DMA request is generated to read out the secondary result (RES2). When all results have been read, the RRDY flag is deasserted.

The DMA read and write channels can be enabled separately. If both channels are enabled, the CORDIC can autonomously perform repeated calculations on a buffer of data without processor intervention.This allows the processor to perform other tasks. The DMA controller is operating in memory-to-peripheral mode for the write channel, and peripheral-to-memory mode for the read channel. Note that the sequence is started by the processor setting the DMAWEN flag. Thereafter the DMA read and write requests are generated as fast as the CORDIC can process the data.

In some cases, the input data may be stored in memory, and the output is transferred at regular intervals to another peripheral, such as a digital-to-analog converter. In this case, the destination peripheral generates a DMA request each time it needs a new data. The DMA controller can directly fetch the next sample from the CORDIC_RDATA register (in this case the DMA controller is operating in memory-to-peripheral mode, even though the source is a peripheral register). The act of reading the result allows the CORDIC to start a new calculation, which in turn generates a DMA write channel request, and the DMA controller transfers the next input value to the CORDIC_WDATA register. The DMA write channel is enabled (DMAWEN = 1), but the read channel must not be enabled.

In a similar way, data coming from another peripheral, such as an ADC, can be transferred directly to the CORDIC_WDATA register (in peripheral-to-memory mode). The DMA write channel must not be enabled. The CORDIC processes the input data and generates a DMA read request when complete, if DMAREN = 1. The DMA controller then transfers the result from CORDIC_RDATA register to memory (peripheral-to-memory mode).

Note: No DMA request is generated to program the CORDIC_CSR register. DMA mode is therefore only useful when repeatedly performing the same function with the same settings. Note too that the scale factor cannot be changed during a series of DMA transfers.

Note: Each DMA request must be acknowledged, as a result of the DMA performing an access to the CORDIC_WDATA or CORDIC_RDATA register. If an extraneous access to the relevant register occurs before this, the acknowledge is asserted prematurely, and could block the DMA channel. Therefore, when the DMA read channel is enabled, CPU access to the CORDIC_RDATA register must be avoided. Similarly, the processor must avoid accessing the CORDIC_WDATA register when the DMA write channel is enabled.

28.4 CORDIC registers

The CORDIC registers can only be accessed in 32-bit word format.

28.4.1 CORDIC memory map

Table 323. CORDIC register memory map

Offset	Register name
0x00	*CORDIC control/status register (CORDIC_CSR)*
0x04	*CORDIC argument register (CORDIC_WDATA)*
0x8	*CORDIC result register (CORDIC_RDATA)*

28.4.2 CORDIC control/status register (CORDIC_CSR)

Address offset: 0x00

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARG SIZE	RES SIZE	NARGS	NRES	DMA WEN	DMA REN	IEN
r									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SCALE[2:0]			PRECISION[3:0]				FUNC[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RRDY**: Result ready flag

0: No new result in output register

1: CORDIC_RDATA register contains new data.

This bit is set by hardware when a CORDIC operation completes. It is reset by hardware when the CORDIC_RDATA register is read (NRES+1) times.

When this bit is set, if the IEN bit is also set, the CORDIC interrupt is asserted. If the DMAREN bit is set, a DMA read channel request is generated. While this bit is set, no new calculation is started.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **ARGSIZE**: Width of input data

0: 32-bit

1: 16-bit

ARGSIZE selects the number of bits used to represent input data.

If 32-bit data is selected, the CORDIC_WDATA register expects arguments in q1.31 format.

If 16-bit data is selected, the CORDIC_WDATA register expects arguments in q1.15 format. The primary argument (ARG1) is written to the least significant half-word, and the secondary argument (ARG2) to the most significant half-word.

Bit 21 **RESSIZE**: Width of output data

0: 32-bit

1: 16-bit

RESSIZE selects the number of bits used to represent output data.

If 32-bit data is selected, the CORDIC_RDATA register contains results in q1.31 format.

If 16-bit data is selected, the least significant half-word of CORDIC_RDATA contains the primary result (RES1) in q1.15 format, and the most significant half-word contains the secondary result (RES2), also in q1.15 format.

Bit 20 **NARGS**: Number of arguments expected by the CORDIC_WDATA register

0: Only one 32-bit write (or two 16-bit values if ARGSIZE = 1) is needed for the next calculation.

1: Two 32-bit values must be written to the CORDIC_WDATA register to trigger the next calculation.

Reads return the current state of the bit.

Bit 19 **NRES**: Number of results in the CORDIC_RDATA register

0: Only one 32-bit value (or two 16-bit values if RESSIZE = 1) is transferred to the CORDIC_RDATA register on completion of the next calculation. One read from CORDIC_RDATA resets the RRDY flag.

1: Two 32-bit values are transferred to the CORDIC_RDATA register on completion of the next calculation. Two reads from CORDIC_RDATA are necessary to reset the RRDY flag.

Reads return the current state of the bit.

Bit 18 **DMAWEN**: Enable DMA write channel

0: Disabled. No DMA write requests are generated.

1: Enabled. Requests are generated on the DMA write channel whenever no operation is pending

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 17 **DMAREN**: Enable DMA read channel

0: Disabled. No DMA read requests are generated.

1: Enabled. Requests are generated on the DMA read channel whenever the RRDY flag is set.

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 16 **IEN**: Enable interrupt.
0: Disabled. No interrupt requests are generated.
1: Enabled. An interrupt request is generated whenever the RRDY flag is set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **SCALE[2:0]**: Scaling factor
The value of this field indicates the scaling factor applied to the arguments and/or results. A value n implies that the arguments have been multiplied by a factor 2^{-n}, and/or the results need to be multiplied by 2^{n}. Refer to *Section 28.3.2* for the applicability of the scaling factor for each function and the appropriate range.

Bits 7:4 **PRECISION[3:0]**: Precision required (number of iterations)
0: reserved
1 to 15: (Number of iterations)/4
To determine the number of iterations needed for a given accuracy refer to *Table 322*.
Note that for most functions, the recommended range for this field is 3 to 6.

Bits 3:0 **FUNC[3:0]**: Function
0: Cosine
1: Sine
2: Phase
3: Modulus
4: Arctangent
5: Hyperbolic cosine
6: Hyperbolic sine
7: Arctanh
8: Natural logarithm
9: Square Root
10 to 15: reserved

28.4.3 CORDIC argument register (CORDIC_WDATA)

Address offset: 0x04

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARG[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARG[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ARG[31:0]**: Function input arguments

This register is programmed with the input arguments for the function selected in the CORDIC_CSR register FUNC field.

If 32-bit format is selected (CORDIC_CSR.ARGSIZE = 0) and two input arguments are required (CORDIC_CSR.NARGS = 1), two successive writes are required to this register. The first writes the primary argument (ARG1), the second writes the secondary argument (ARG2).

If 32-bit format is selected and only one input argument is required (NARGS = 0), only one write is required to this register, containing the primary argument (ARG1).

If 16-bit format is selected (CORDIC_CSR.ARGSIZE = 1), one write to this register contains both arguments. The primary argument (ARG1) is in the lower half, ARG[15:0], and the secondary argument (ARG2) is in the upper half, ARG[31:16]. In this case, NARGS must be set to 0.

Refer to *Section 28.3.2* for the arguments required by each function, and their permitted range.

When the required number of arguments has been written, the CORDIC evaluates the function designated by CORDIC_CSR.FUNC using the supplied input arguments, provided any previous calculation has completed. If a calculation is ongoing, the ARG1 and ARG 2 values are held pending until the calculation is completed and the results read. During this time, a write to the register cancels the pending operation and overwrite the argument data.

28.4.4 CORDIC result register (CORDIC_RDATA)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RES[31:0]**: Function result

If 32-bit format is selected (CORDIC_CSR.RESSIZE = 0) and two output values are expected (CORDIC_CSR.NRES = 1), this register must be read twice when the RRDY flag is set. The first read fetches the primary result (RES1). The second read fetches the secondary result (RES2) and resets RRDY.

If 32-bit format is selected and only one output value is expected (NRES = 0), only one read of this register is required to fetch the primary result (RES1) and reset the RRDY flag.

If 16-bit format is selected (CORDIC_CSR.RESSIZE = 1), this register contains the primary result (RES1) in the lower half, RES[15:0], and the secondary result (RES2) in the upper half, RES[31:16]. In this case, NRES must be set to 0, and only one read performed.

A read from this register resets the RRDY flag in the CORDIC_CSR register.

29 Analog-to-digital converters (ADC)

29.1 Introduction

This section describes the implementation of up to 5 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).
- ADC3 and ADC4 are tightly coupled and can operate in dual mode (ADC3 is master).
- ADC5 is controlled independently.

Each ADC consists of a 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 19 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of ADC is stored in a left-aligned or right-aligned 16-bit data register.

ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows to improve analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

29.2 ADC main features

- High-performance features
  - Up to 5 ADCs, out of which four of them (in pairs) can operate in dual mode:
    ADC1 is connected to 15 external channels + 3 internal channels
    ADC2 is connected to 15 external channels + 3 internal channels
    ADC3 is connected to 15 external channels + 3 internal channels
    ADC4 is connected to 15 external channels + 3 internal channels
    ADC5 is connected to 12 external channels + 6 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Manage single-ended or differential inputs
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Four dedicated data registers for the injected channels
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8 bits
- Data preconditioning
  - Offset compensation
- Analog input channels
  - External analog inputs (per ADC):
    Up to 5 fast channels from GPIO pads
    Up to 10 slow channels from GPIO pads
  - 2 channels for the internal temperature sensor
  - 1 channel for the internal reference voltage (V_{REF})

- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
  - Watchdog can perform filtering to ignore out-of-range data
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

Figure 235 shows the block diagram of one ADC.

29.3 ADC implementation

Table 324. ADC features

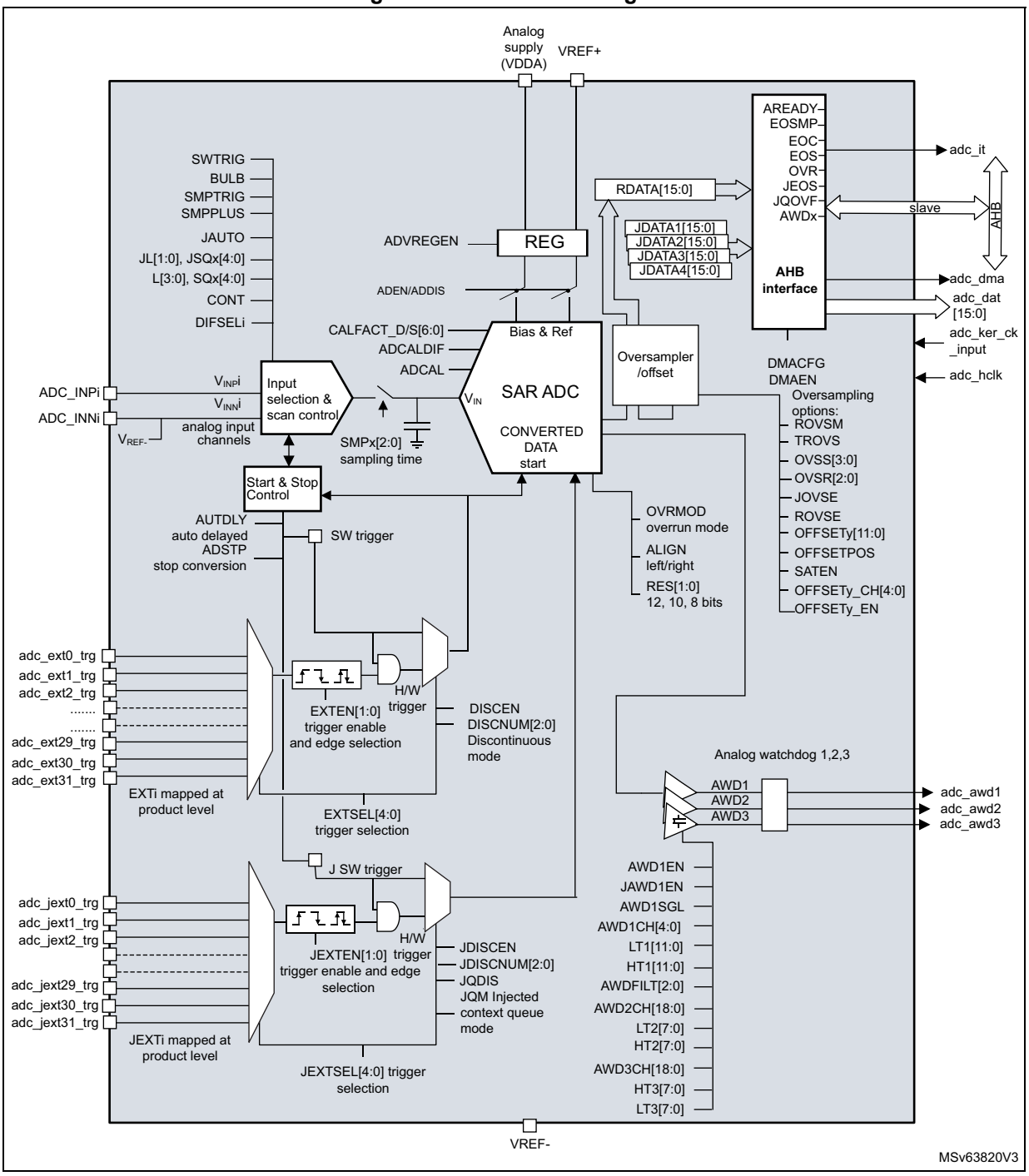
ADC modes/features	ADC1, ADC2	ADC3, ADC4, ADC5
Resolution	12 bits	12 bits
Maximum sampling speed	2.5 Msps (12-bit resolution)	2.5 Msps (12-bit resolution)
Dual mode operation	Yes	Yes
Hardware offset calibration	Yes	Yes
Hardware linearity calibration	Yes	Yes
Single-end input	Yes	Yes
Differential input	Yes	Yes
Injected channel conversion	Yes	Yes
Oversampling	up to x256	up to x256
Data register	16 bits	16 bits
DMA support	Yes	Yes
Parallel data output to DFSDM	No	No
Offset compensation	Yes	Yes
Gain compensation	No	No
Number of Analog watchdog	3	3

29.4 ADC functional description

29.4.1 ADC block diagram

Figure 235 shows the ADC block diagram and *Table 325* gives the ADC pin description.

Figure 235. ADC block diagram



29.4.2 ADC pins and internal signals

Table 325. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply and positive reference voltage for the ADC
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, analog reference positive	The higher/positive reference voltage for ADC.
VREF−	Input, analog reference negative	The lower/negative reference voltage for ADC.
ADC_INNi/INPi	Negative/positive external analog input signals	16 negative/positive external analog input channels (coming from GPIOs). Refer to the list of the ADC input channel connections in the *Chapter 5: Device configuration* for details.

Table 326. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{INP}i$	Positive analog input channels	Positive internal analog input channels connected either to ADCx_INP*i* external channels or to internal channels.
$V_{INN}i$	Negative analog input channels	Negative internal analog input channels connected either to ADCx_INN*i* external channels or to internal channels
adc_ext_trgi	Inputs	ADC external trigger inputs for regular conversions.These inputs are shared between the ADC master and the ADC slave.
adc_jext_trgi	Inputs	ADC external trigger inputs for the injected conversions. These inputs are shared between the ADC master and the ADC slave.
adc_awdx	Output	Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)
adc_ker_ck_input	Output	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request
adc_dat[15:0]	Output	ADC data outputs

29.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The ADC input clock can be selected between two different clock sources (see *Figure 236: ADC clock scheme*):

1. The ADC clock can be a specific clock source (adc_ker_ck_input), independent and asynchronous with the AHB clock.

   Refer to *Chapter 11: Reset and clock control (RCC)* for more information on how to generate the ADC dedicated clock. To select this scheme, CKMODE[1:0] bits of ADCx_CCR register must be set to 00.
2. The ADC clock can be derived from the AHB clock interface divided by a programmable factor of 2 or 4. To select this scheme, CKMODE[1:0] bits of ADCx_CCR must be different from 00. The programmable divider factor can be configured through CKMODE[1:0] bits of ADCx_CCR.

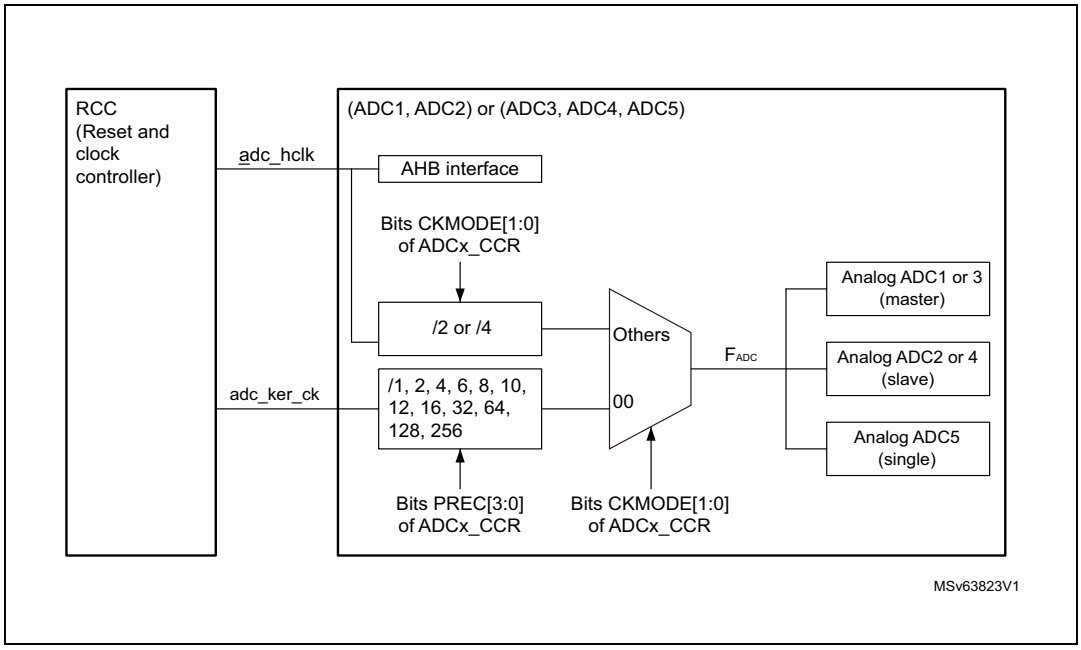
Option **1** has the advantage of achieving the maximum ADC clock frequency, whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256, using the prescaler configured with bits PRESC[3:0] in the ADC_CCR register.

Option **2** has the advantage of bypassing the clock domain resynchronizations. This can be useful when ADC is triggered by a timer and if the application requires that ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

The clock configured through CKMODE[1:0] bits must be compliant with the operating frequency specified in the device datasheet.

Note: when selecting the option 2, the choice of prescaling factor CKMODE[1:0] requires reducing the AHB clock frequency (FSYS).
Example: to satisfy the ADC clock limits, when CKMODE[1:0] = 11 the AHB frequency cannot exceed 150 MHz.

Figure 236. ADC clock scheme



Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{adc_hclk} \geq F_{ADC} / 4$ if the resolution of all channels are 12-bit or 10-bit
- $F_{adc_hclk} \geq F_{ADC} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)
- $F_{adc_hclk} \geq F_{ADC} / 2$ if there are some channels with resolutions equal to 6-bit

29.4.4 Slave AHB interface

ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

29.4.5 ADC voltage regulator (ADVREGEN)

It is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN = 1 into ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG_STUP}$) before launching a calibration or enabling ADC. This delay must be implemented by software.

For the startup time of the ADC voltage regulator, refer to device datasheet for $T_{ADCVREG_STUP}$ parameter.

After ADC operations are complete, ADC can be disabled (ADEN = 0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN = 0.

Note: When the internal voltage regulator is disabled (ADVREGEN = 0), the internal analog calibration is kept.

29.4.6 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be written while ADC is disabled (ADEN = 0). Note that the DIFSEL[i] bits corresponding to single-ended channels are always programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and V_{SSA} (analog supply ground).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

The output data for the differential mode is an unsigned data. When $V_{INP[i]}$ equals V_{REF-}, $V_{INN[i]}$ equals V_{REF+} and the output data is 0x000 (12-bit resolution mode). When $V_{INP[i]}$ equals V_{REF+}, $V_{INN[i]}$ equals V_{REF-} and the output data is 0xFFF.

$$\text{Converted value} = \frac{\text{ADC_Full_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When ADC is configured as differential mode, both inputs must be biased at (V_{REF+}) /2 voltage.

The input signals are supposed to be differential (common mode voltage must be fixed).

Internal channels (such as V_{REF}) are used in single-ended mode only.

For a complete description of how the input channels are connected for each ADC, refer to *Chapter 5: Device configuration*.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage $V_{INN[i]}$ is connected to another channel. As a consequence, this channel is no longer usable in Single-ended mode or in differential mode and must never be configured to be converted. Some channels are shared between ADC1/ADC2/ADC3/ADC4/ADC5: this can make the channel on the other ADC unusable. Only exception is interleaved mode for ADC master and the slave.

29.4.7 Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of ADC. During the procedure, ADC calculates a calibration factor which is 7-bit wide and which is applied internally to ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF = 0 before launching a calibration which is applied for single-ended input conversions.
- Write ADCALDIF = 1 before launching a calibration which is applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when ADC is disabled (when ADEN = 0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon as the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADC_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if ADC is disabled (ADEN = 0). However, if ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling ADC.

The internal analog calibration is lost each time the power of ADC is removed. In this case, to avoid spending time recalibrating ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

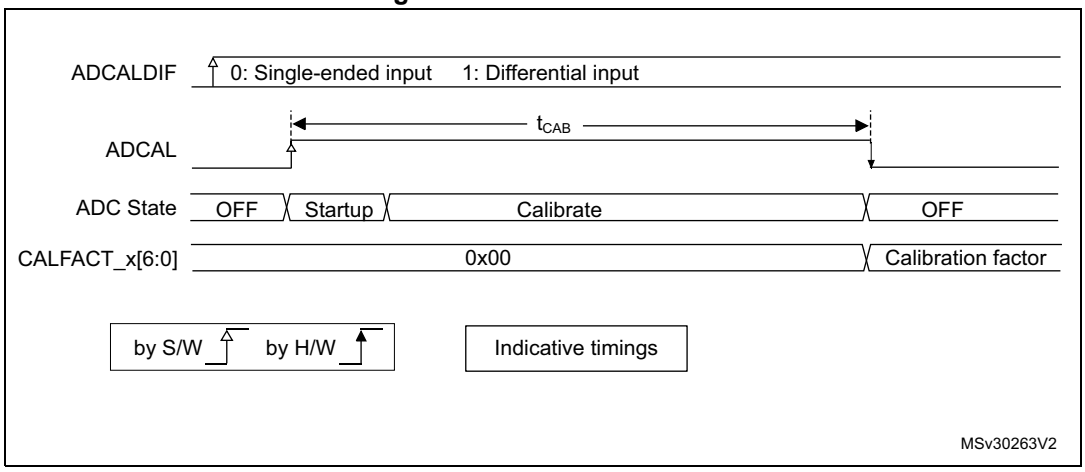
The calibration factor can be written if ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when V_{REF+} voltage changed more than 10%.

Note: if the calibration procedure is skipped (bit ADCAL = 0), the offset error, stored into the analog ADC, is equal to 0x40 in 12-bit configuration mode (0x10 in 10-bit configuration mode).
Each write of the bit field CALFACT_S or CALFACT_D is done with the new calibration factors, it is applied once a new conversion is launched.

Software procedure to calibrate ADC

1. Ensure ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by setting ADCALDIF = 0 (single-ended input) or ADCALDIF = 1 (differential input).
4. Set ADCAL = 1.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from ADC_CALFACT register.

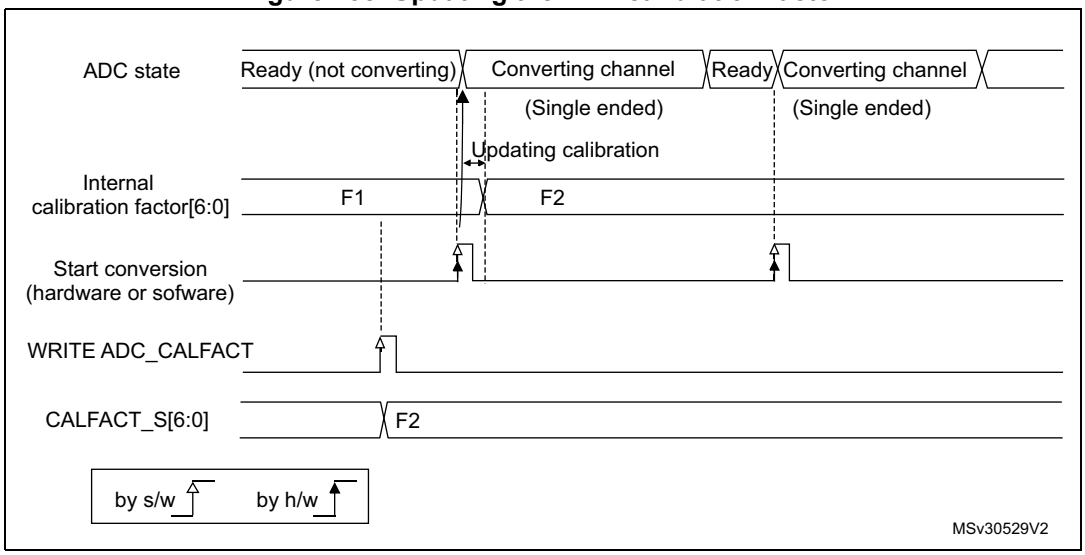
Figure 237. ADC calibration



Software procedure to re-inject a calibration factor into ADC

1. Ensure ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 238. Updating the ADC calibration factor

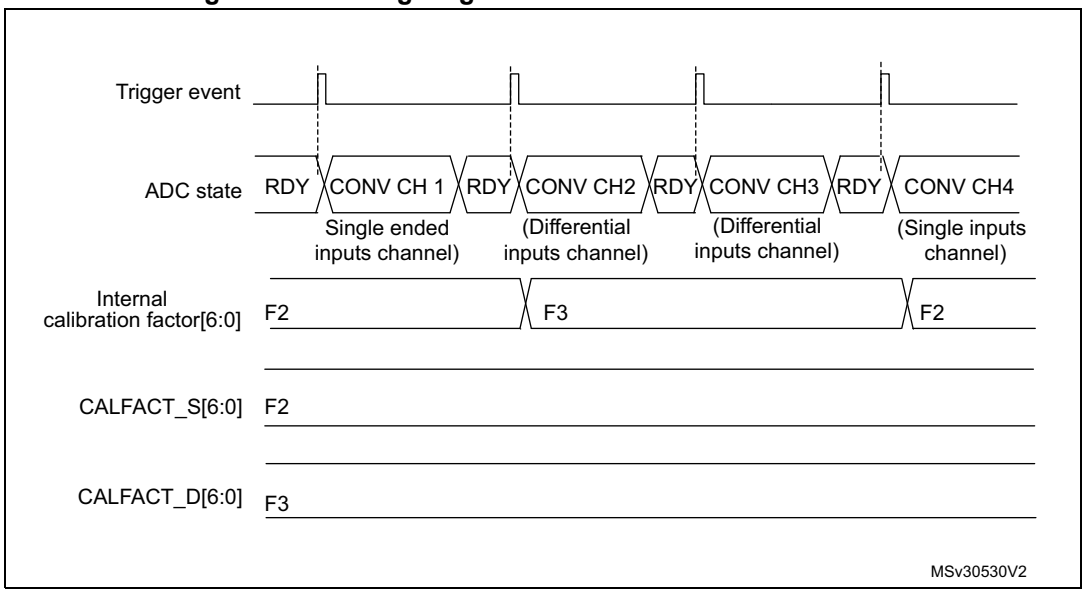


Converting single-ended and differential analog inputs with a single ADC

If ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable ADC.
2. Calibrate ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT_S[6:0].
3. Calibrate ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT_D[6:0].
4. Enable ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration is automatically injected into the analog ADC.

Figure 239. Mixing single-ended and differential channels



29.4.8 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in *Section 29.4.5: ADC voltage regulator (ADVREGEN)*).

Once ADVREGEN = 1, ADC can be enabled and ADC needs a stabilization time of t_{STAB} before it starts converting accurately, as shown in *Figure 240*. Two control bits enable or disable ADC:

- ADEN = 1 enables ADC. The flag ADRDY is set once ADC is ready for operation.
- ADDIS = 1 disables ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART = 1 (refer to *Section 29.4.17: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)*) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART = 1 or when an external injected trigger event occurs, if injected triggers are enabled.

Software procedure to enable ADC

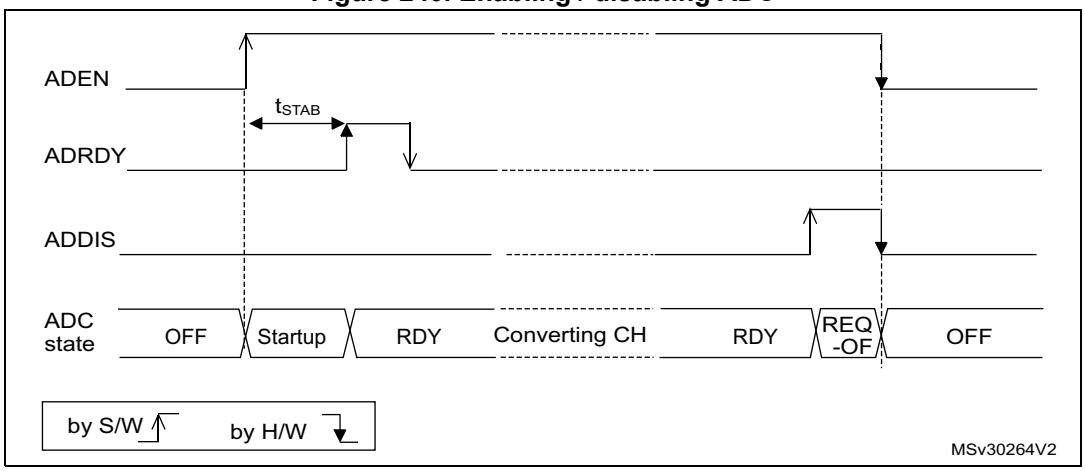
1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set ADEN = 1.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE = 1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

Caution: ADEN bit cannot be set when ADCAL is set and during four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).

Software procedure to disable ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP = 1 and JADSTP = 1 and then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS = 1.
3. If required by the application, wait until ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 240. Enabling / disabling ADC



29.4.9 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the DIFSEL[i] control bits in the ADC_DIFSEL register and the control bits ADCAL and ADEN in the ADC_CR register, only if ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if ADC is enabled and there is no pending request to disable ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if ADC is enabled (ADEN = 1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if ADC is enabled (ADEN = 1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).
- ADC_TRy registers can be modified when an analog-to-digital conversion is ongoing (refer to *Section 29.4.27: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*for details).

The software is allowed to write the ADSTP or JADSTP control bits of the ADC_CR register only if ADC is enabled, possibly converting, and if there is no pending request to disable ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC_JSQR at any time, when ADC is enabled (ADEN = 1). Refer to *Section 29.6.1.16: ADC injected sequence register (ADC_JSQR)* for additional details.

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may result in an unknown state. To recover from this situation, ADC must be disabled (clear ADEN = 0 as well as all the bits of ADC_CR register).

29.4.10 Channel selection (SQRx, JSQRx)

The ADC features up to 19 multiplexed channels per ADC, out of which:

- Up to 16 analog inputs coming from GPIO pads (ADC_INP/INN[i]) depending on the products, not all of them are available on GPIO pads.
- Each ADC is connected to 3 internal analog inputs.

  To convert one of the internal analog channels, the corresponding analog sources must be selected as reported in *Table 23: ADCx connectivity (x = 1..5)* (temperature sensor [PTAT, CTAT] or reference voltages selected by the SYS_CFG.ST_VREF_SEL2 register).

Refer to the table ADC interconnection in *Chapter 5: Device configuration* for the connection of the above internal analog inputs to external ADC pins or internal signals.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_INP/INN3, ADCx_INP/INN8, ADCx_INP/INN2, ADCx_INN/INP2, ADCx_INP/INN0, ADCx_INP/INN2, ADCx_INP/INN2, ADCx_INP/INN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP = 1 (refer to *Section 29.4.16: Stopping an ongoing conversion (ADSTP, JADSTP)*).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set to 1 (injected conversions ongoing) only when the context queue is enabled (JQDIS = 0 in ADC_CFGR register). Refer to *Section 29.4.20: Queue of context for injected conversions*

29.4.11 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

T_{CONV} = Sampling time + 12.5 ADC clock cycles

Example:

With F_{ADC} = 40 MHz and a sampling time of 2.5 ADC clock cycles with ADC_SMPR1.SMPPLUS = 1:

T_{CONV} = (2.5 +1 + 12.5) ADC clock cycles = 16 ADC clock cycles = 400 ns

It means 2.5 MSPS in single mode.

ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the device datasheet.

Bulb sampling mode

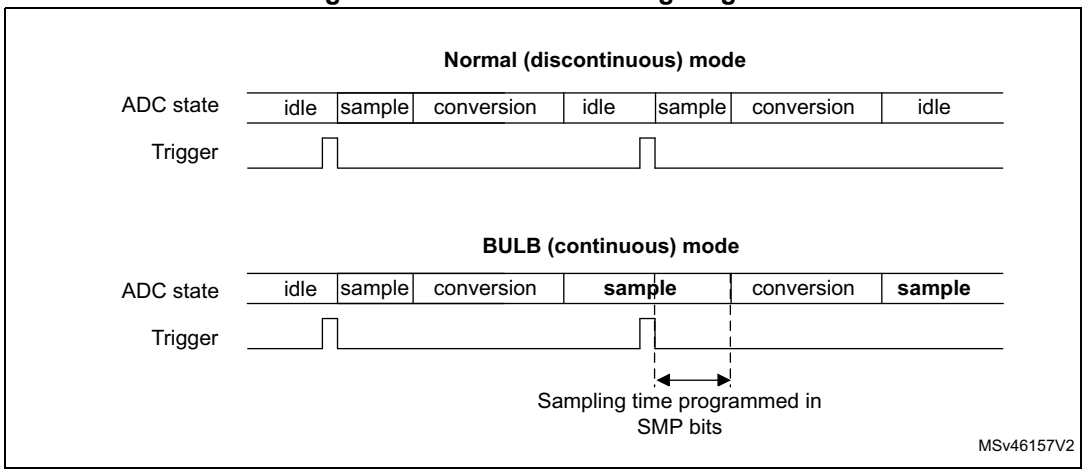
When the BULB bit is set in ADC register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPR1 register. The very first ADC conversion, after ADC is enabled, is performed with the sampling time programmed in SMP bits. The Bulb mode is effective starting from the second conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the device datasheet).

The Bulb mode is neither compatible with the continuous conversion mode nor with the injected channel conversion.

When the BULB bit is set, it is not allowed to set SMPTRIG bit in ADC_CFGR2.

Figure 241. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed through SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected, each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

When a software trigger is selected, the software trigger is not the ADSTART bit in ADC_CR but the SWTRIG bit. SWTRIG bit has to be set to start the sampling period, and the SWTRIG bit has to be cleared to end the sampling period and start the conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the device datasheet).

This mode is neither compatible with the continuous conversion mode, nor with the injected channel conversion.

When SMPTRIG bit is set, it is not allowed to set BULB bit.

SMPPLUS control bit

The SMPPLUS bit adds an additional clock to the sampling time only when it is set to 2.5 ADC clock cycles in ADC_SMPR1 and ADC_SMPR2.

The suggested use case for 2.5 MSPS in single mode requires the setting of the ADC_SMPR1.SMPPLUS bit to 1.

29.4.12 Single conversion mode (CONT = 0)

In Single conversion mode, ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

29.4.13 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both Discontinuous mode and Continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in Continuous mode (using JAUTO bit), refer to Auto-injection mode section).

29.4.14 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART = 1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN is not equal to 0x0

Software starts ADC injected conversions by setting JADSTART = 1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN is not equal to 0x0

Note: In auto-injection mode (JAUTO = 1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure ADC while ADSTART = 0 and JADSTART = 0 are both true, indicating that ADC is idle.

ADSTART is cleared by hardware:

- In Single mode with software regular trigger (CONT = 0, EXTSEL = 0x0)
  - At any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT = x, EXTSEL = x)
  - After execution of the ADSTP procedure asserted by the software.

Note: In Continuous mode (CONT = 1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in Single mode (CONT = 0 and EXTSEL # 0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:

- In Single mode with software injected trigger (JEXTSEL = 0x0)
  - At any end of injected conversion sequence (JEOS assertion) or at any end of subgroup processing if JDISCEN = 1
- in all cases (JEXTSEL = x)
  - After execution of the JADSTP procedure asserted by the software.

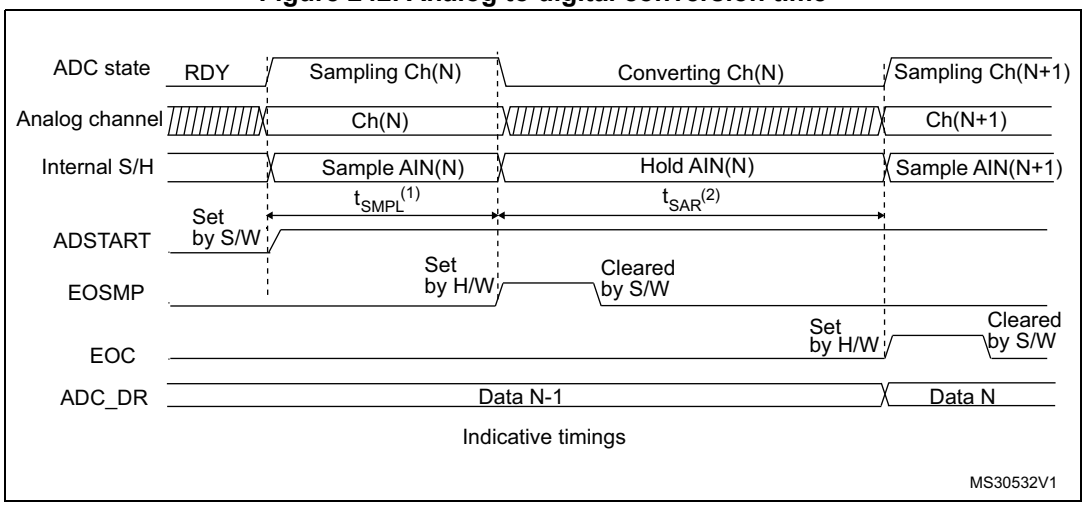
Note: When the software trigger is selected, ADSTART bit must not be set if the EOC flag is still high.

29.4.15 ADC timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$T_{CONV} = T_{SMPL} + T_{SAR} = [2.5\ _{|min} + 12.5\ _{|12bit}] \times T_{ADC_CLK}$
$T_{CONV} = T_{SMPL} + T_{SAR} = 83.33\ ns\ _{|min} + 416.67\ ns\ _{|12bit} = 500.0\ ns$ (for $F_{ADC_CLK} = 30$ MHz)

Figure 242. Analog to digital conversion time



1. T_{SMPL} depends on SMP[2:0].
2. T_{SAR} depends on RES[2:0].

29.4.16 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP = 1 and injected conversions ongoing by setting JADSTP = 1.

Stopping conversions reset the ongoing ADC operation. Then ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching ADC would restart a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the

software must poll ADSTART (or JADSTART) until the bit is reset before assuming ADC is completely stopped.

Note: In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 243. Stopping ongoing regular conversions

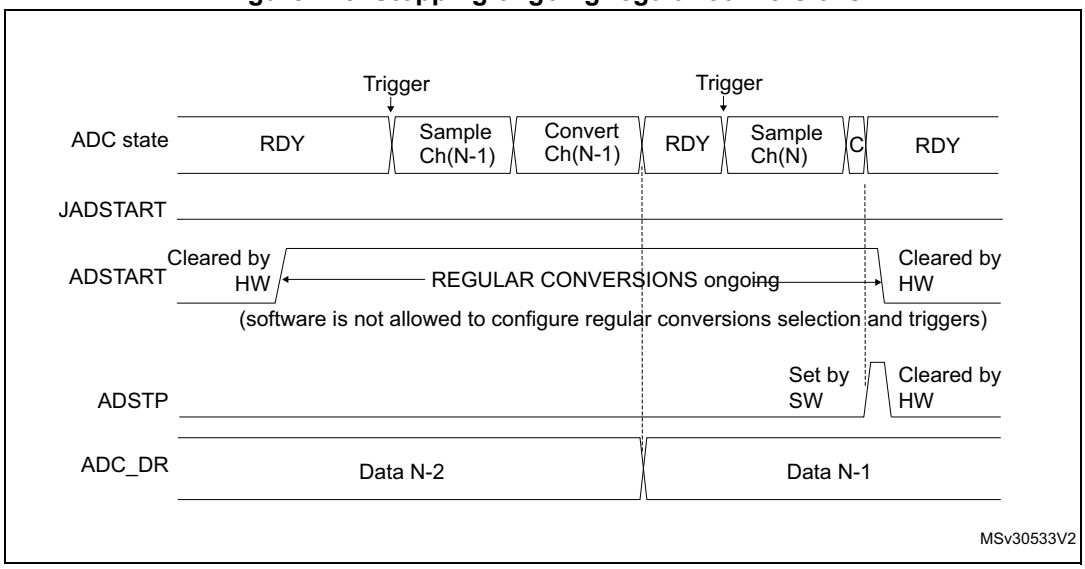
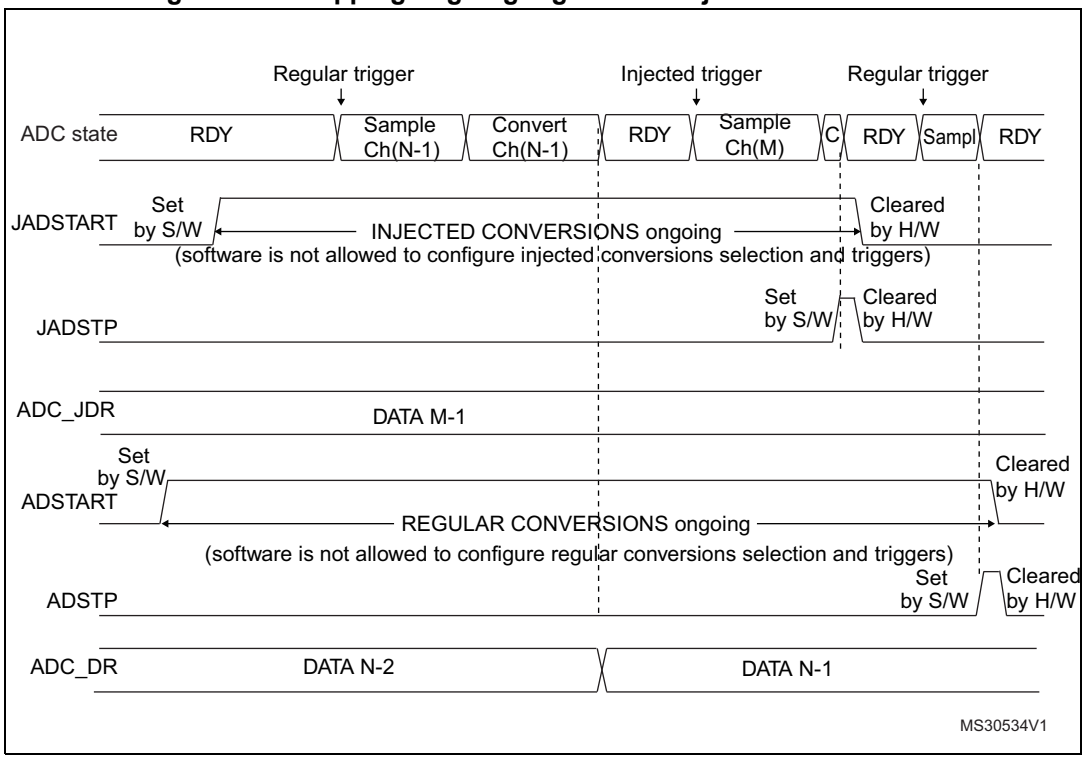


Figure 244. Stopping ongoing regular and injected conversions



29.4.17 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (such as timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS = 0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART = 1 and the injected trigger selection is effective once software has set bit JADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART = 0, any regular hardware triggers which occur are ignored.
- If bit JADSTART = 0, any injected hardware triggers which occur are ignored.

Table 327 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 327. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Table 328. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	– If JQDIS = 1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled – If JQDIS = 0 (Queue enabled), Hardware and software trigger detection disabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

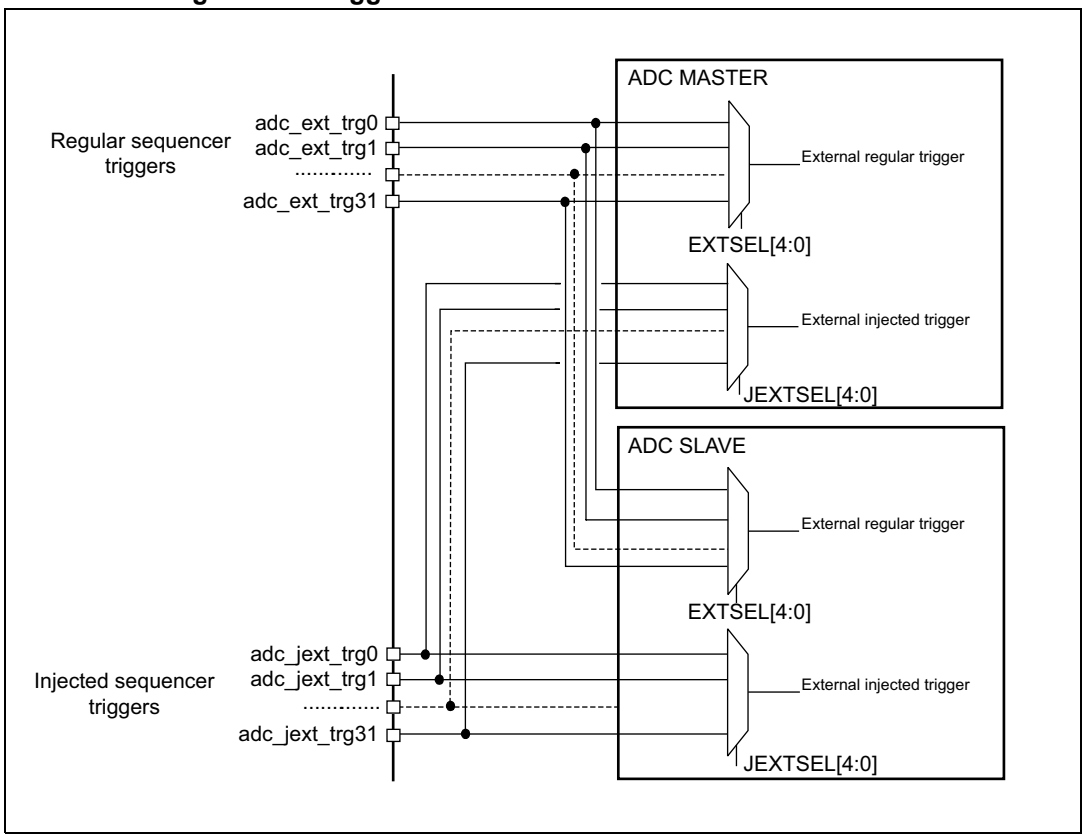
Note: The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS = 0). Refer to Section 29.4.20: Queue of context for injected conversions.

The EXTSEL and JEXTSEL control bits select which out of 32 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly. The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 29.4.20: Queue of context for injected conversions on page 721.

Figure 245. Triggers shared between ADC master and slave



Refer to Table ADC interconnection in *Chapter 5: Device configuration* for the list of all the external triggers that can be used for regular conversion.

29.4.18 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 246* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 2.5 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

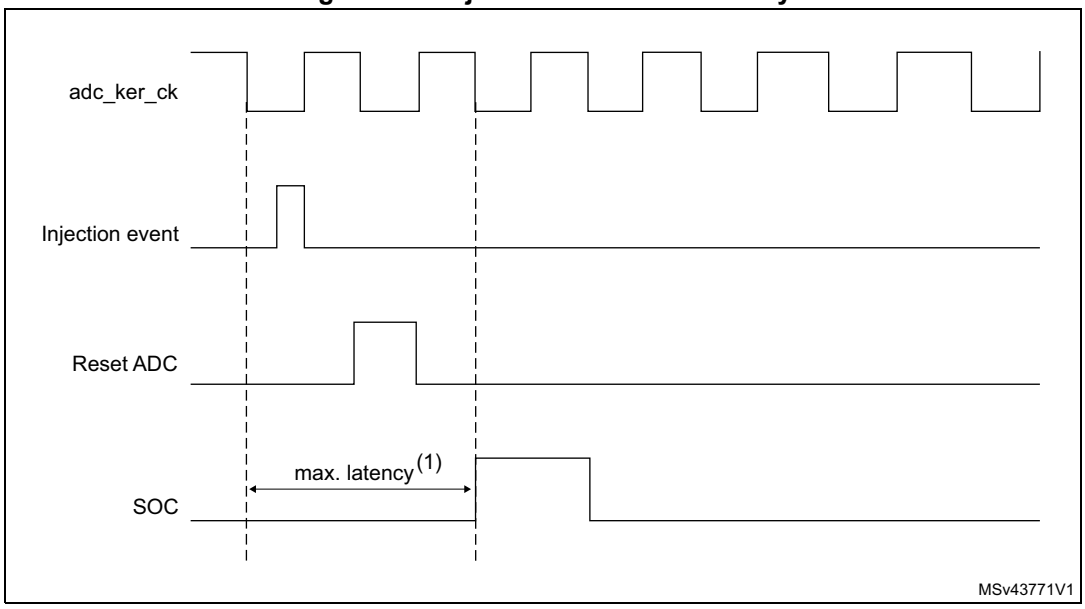
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and Discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (Single-shot mode), the JAUTO sequence is stopped upon DMA Transfer Complete event.

Figure 246. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

29.4.19 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When a regular group is converted in Discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both Discontinuous mode and Continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), ADC behaves as if Continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to Discontinuous mode for regular channels where ‘n’ is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and Discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

29.4.20 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)

All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.

- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  – If JQM = 0, the Queue is empty just after enabling ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence is served according to the last active context.
  – If JQM = 1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP = 1 or when disabling ADC by setting ADDIS = 1:
  – If JQM = 0, the Queue is maintained with the last active context.
  – If JQM = 1, the Queue becomes empty and triggers are ignored.

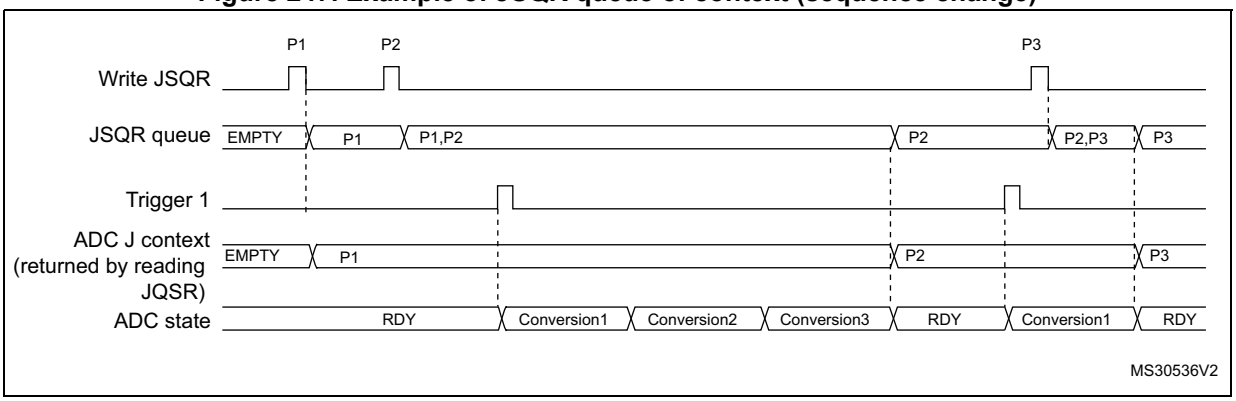
Note: When configured in Discontinuous mode (bit JDISCEN = 1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the Discontinuous mode example below (length = 3 for both contexts):

- *1st trigger, Discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out*
- *2nd trigger, disc. Sequence 1: 2nd conversion.*
- *3rd trigger, Discontinuous. Sequence 1: 3rd conversion.*
- *4th trigger, Discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.*
- *5th trigger, Discontinuous. Sequence 2: 2nd conversion.*
- *6th trigger, Discontinuous. Sequence 2: 3rd conversion.*

Behavior when changing the trigger or sequence context

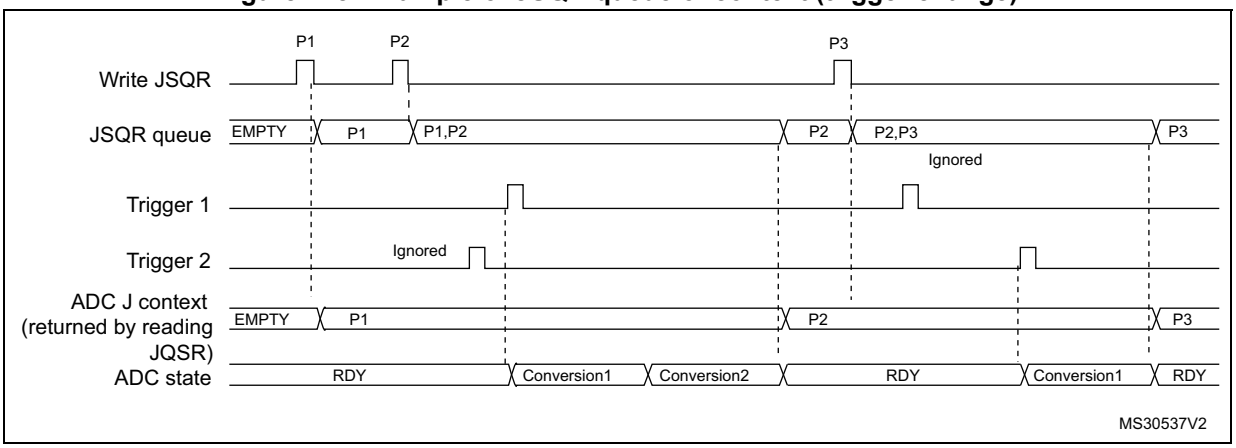
Figure 247 and *Figure 248* show the behavior of the context Queue when changing the sequence or the triggers.

Figure 247. Example of JSQR queue of context (sequence change)



Note: Parameters:
P1: sequence of 3 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 4 conversions, hardware trigger 1

Figure 248. Example of JSQR queue of context (trigger change)

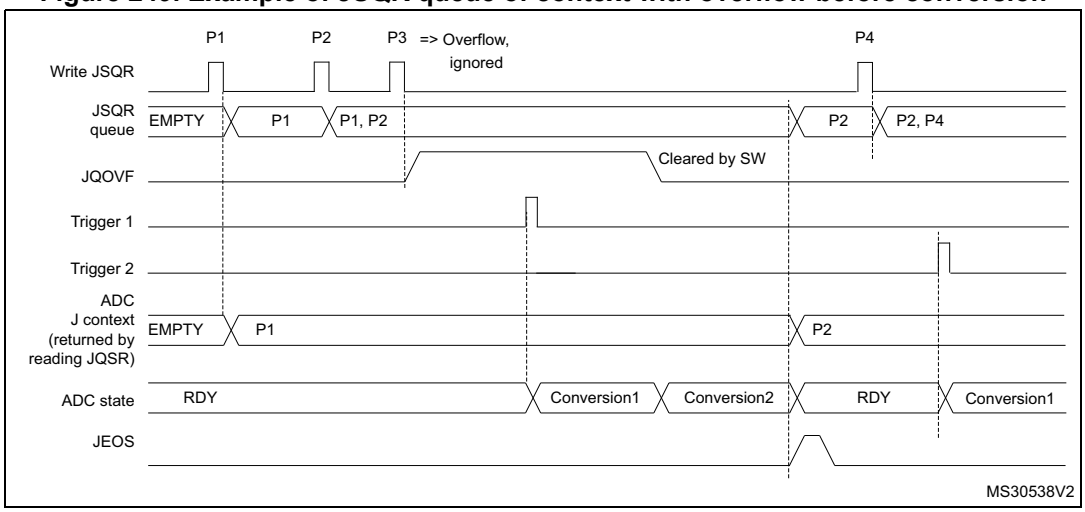


Note: Parameters:
P1: sequence of 2 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 2
P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

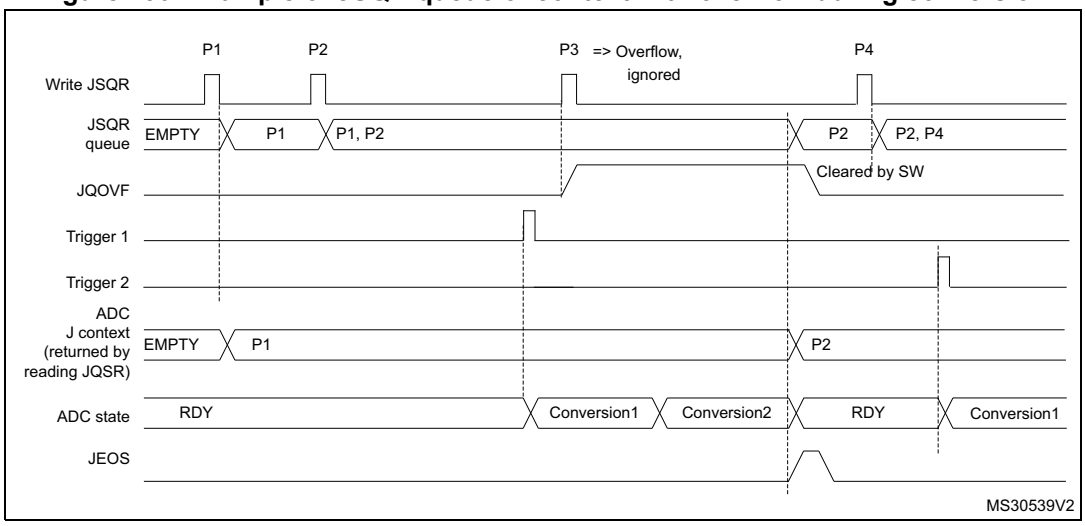
The *Figure 249* and *Figure 250* show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 249. Example of JSQR queue of context with overflow before conversion



Note: Parameters:
P1: sequence of 2 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 2
P3: sequence of 3 conversions, hardware trigger 1
P4: sequence of 4 conversions, hardware trigger 1

Figure 250. Example of JSQR queue of context with overflow during conversion



Note: Parameters:
P1: sequence of 2 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 2
P3: sequence of 3 conversions, hardware trigger 1
P4: sequence of 4 conversions, hardware trigger 1

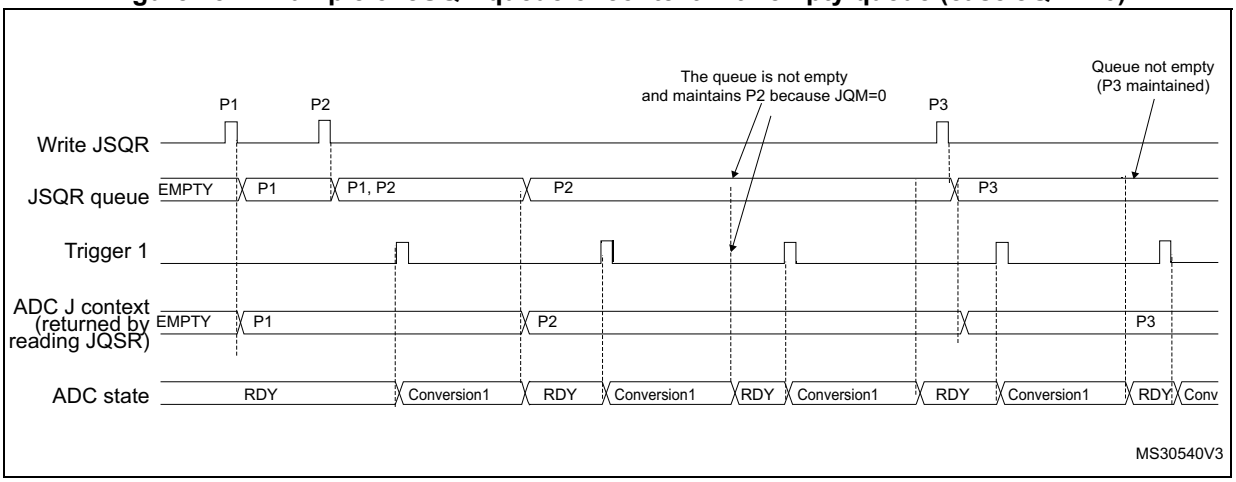
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 251 and *Figure 252* show the behavior of the context Queue when the Queue becomes empty in both cases JQM = 0 or 1.

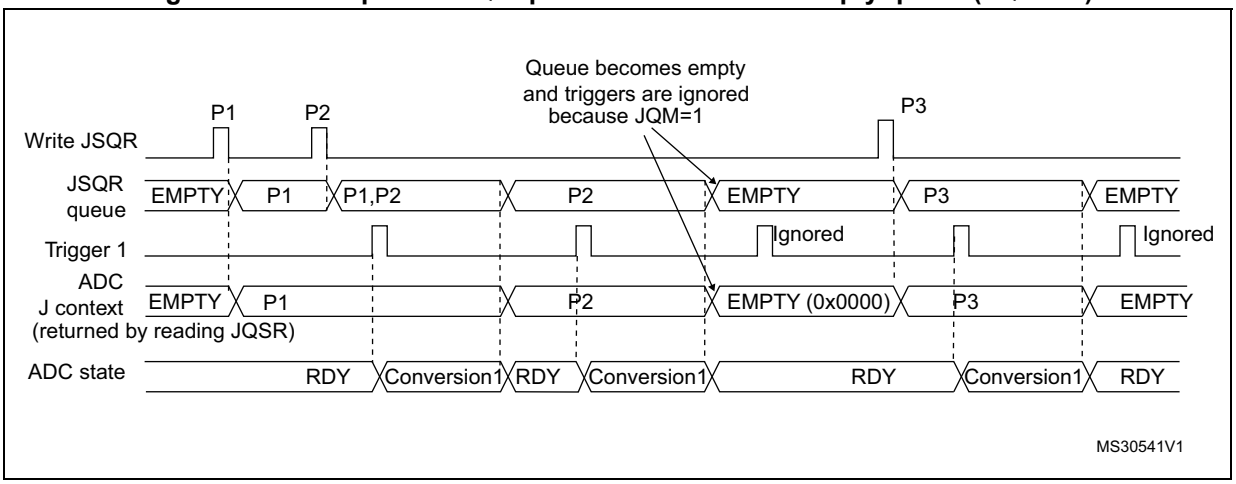
Figure 251. Example of JSQR queue of context with empty queue (case JQM = 0)



Note: Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 252. Example of JSQR queue of context with empty queue (JQM = 1)

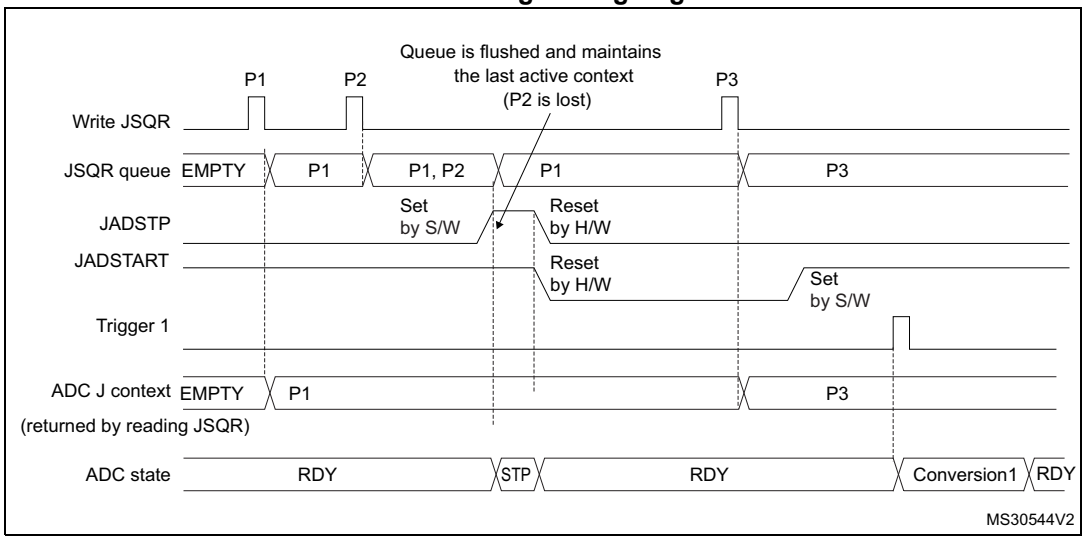


Note: Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

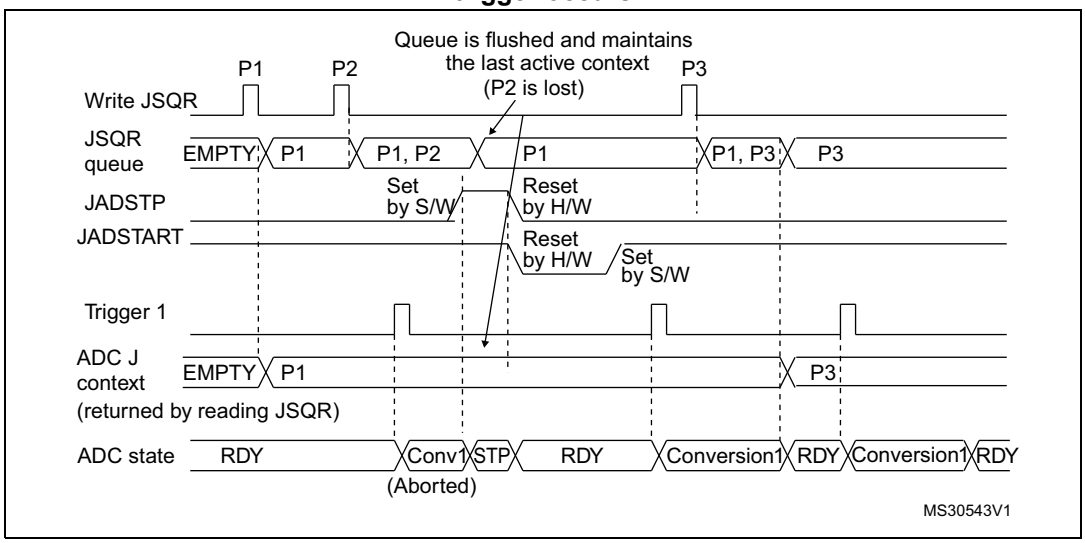
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 253. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion.



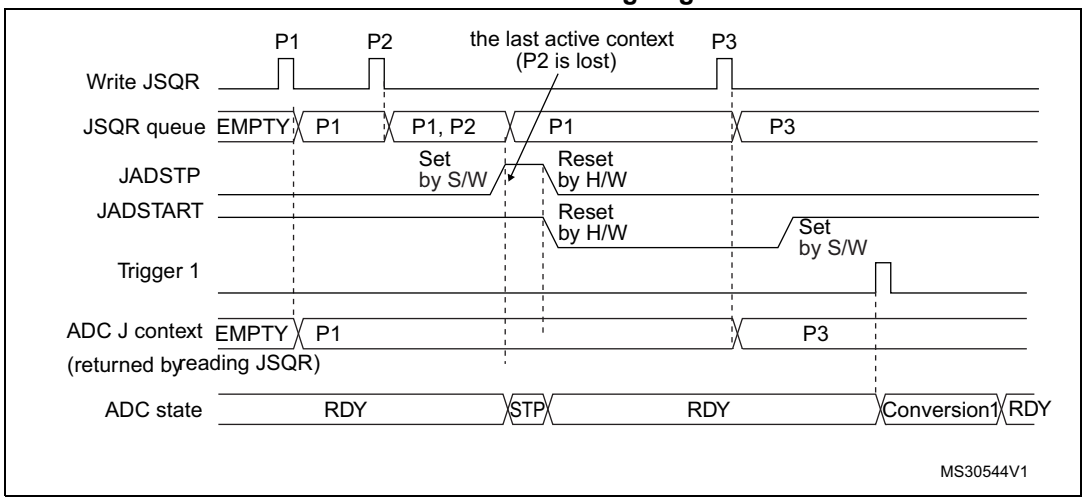
Note: Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 254. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs



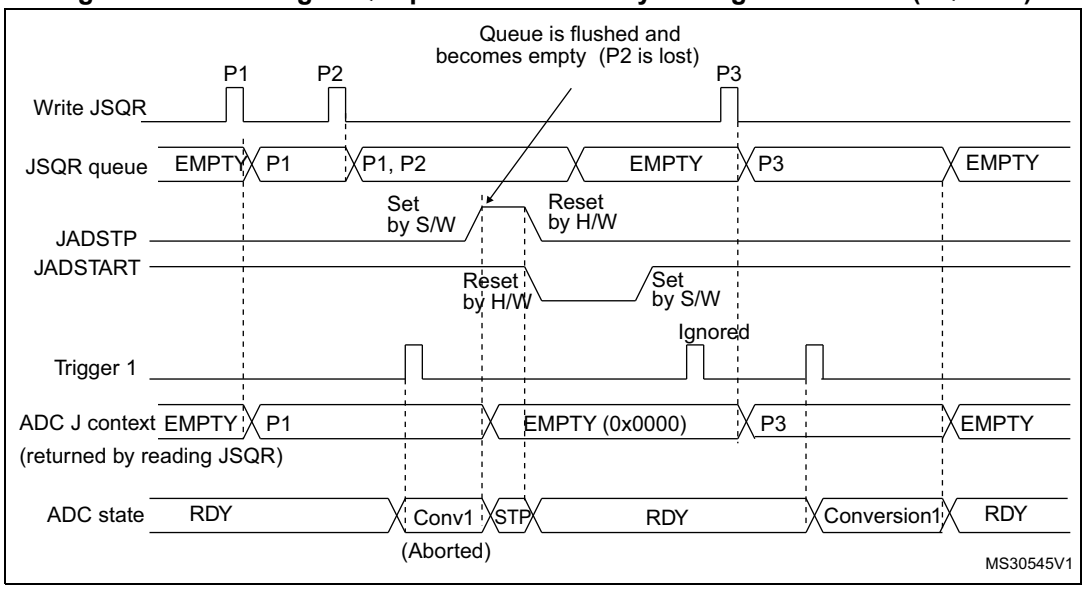
Note: Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 255. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion



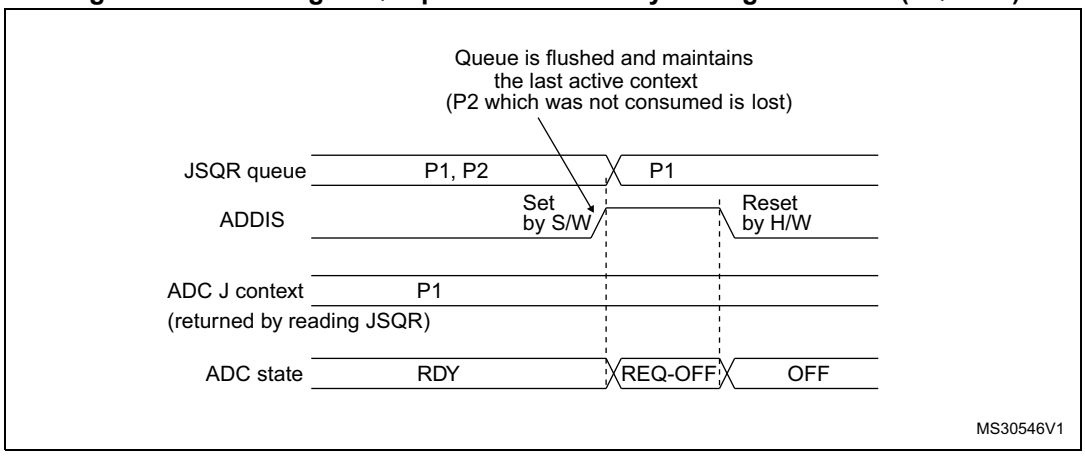
Note: Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 256. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1)



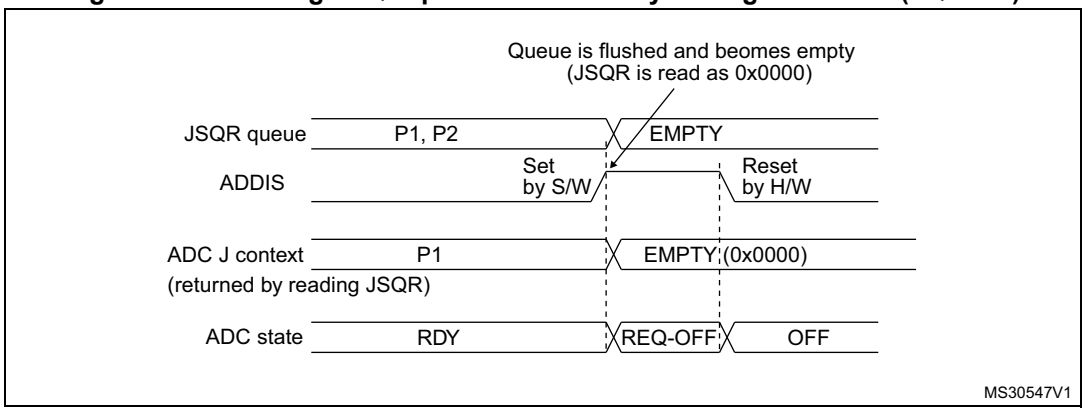
Note: Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 257. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0)



Note: Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 258. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1)



Note: Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Queue of context: Starting ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time ADC is initialized. This procedure is only applicable when JQM bit is reset:

1. Write a dummy JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion).
2. Set JADSTART.
3. Set JADSTP.
4. Wait until JADSTART is reset.
5. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS = 1 into the ADC_CFGR register.

29.4.21 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. *Figure 263*, *Figure 264*, *Figure 265* and *Figure 266* show the conversion result format with respect to the resolution as well as to the data alignment.

A lower resolution allows a faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 329*.

Table 329. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at F_{ADC} = 30 MHz	T_{CONV} (ADC clock cycles) (with Sampling Time = 2.5 ADC clock cycles)	T_{CONV} (ns) at F_{ADC} = 30 MHz
12	12.5 ADC clock cycles	416.67 ns	15 ADC clock cycles	500.0 ns
10	10.5 ADC clock cycles	350.0 ns	13 ADC clock cycles	433.33 ns
8	8.5 ADC clock cycles	203.33 ns	11 ADC clock cycles	366.67 ns
6	6.5 ADC clock cycles	216.67 ns	9 ADC clock cycles	300.0 ns

29.4.22 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

29.4.23 End of conversion sequence (EOS, JEOS)

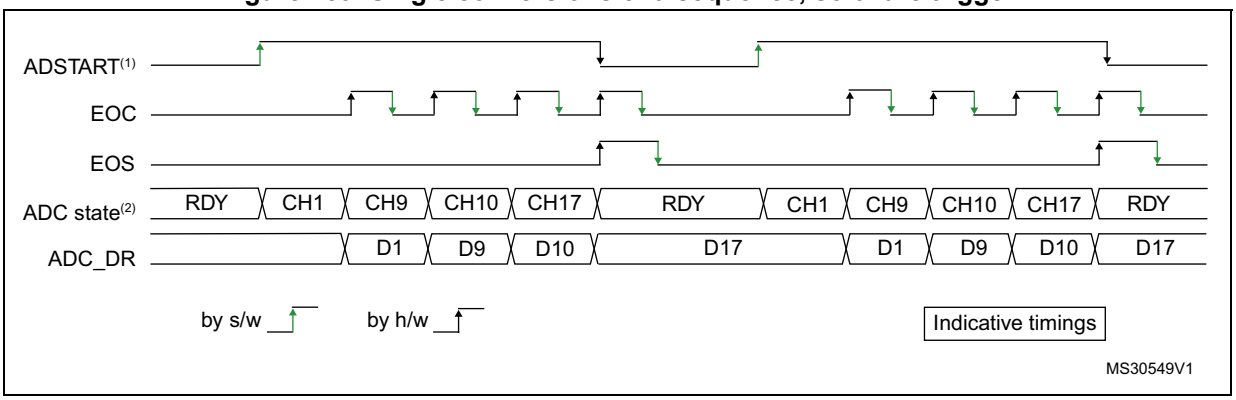
ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

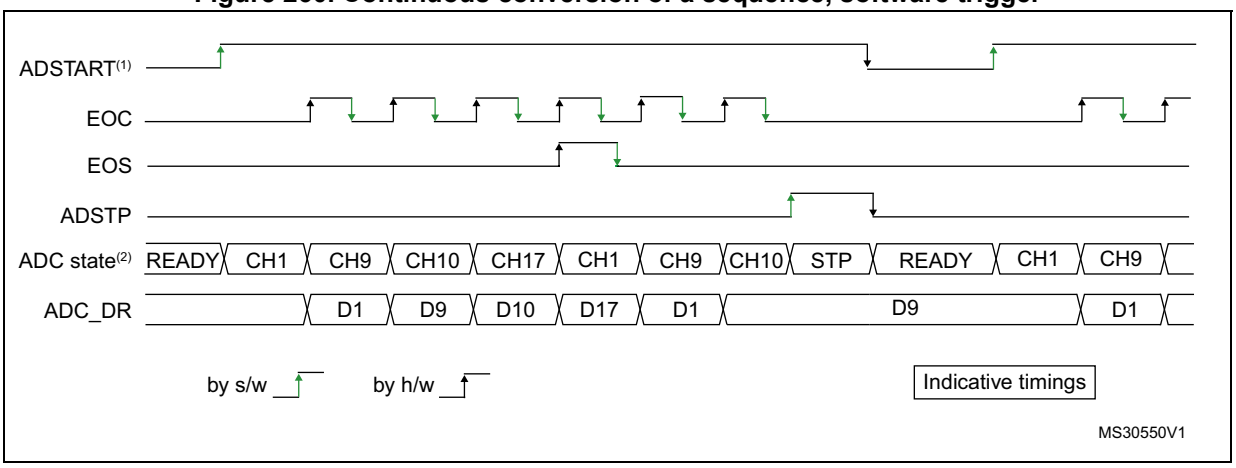
29.4.24 Timing diagrams example (Single/Continuous modes, hardware/software triggers)

Figure 259. Single conversions of a sequence, software trigger



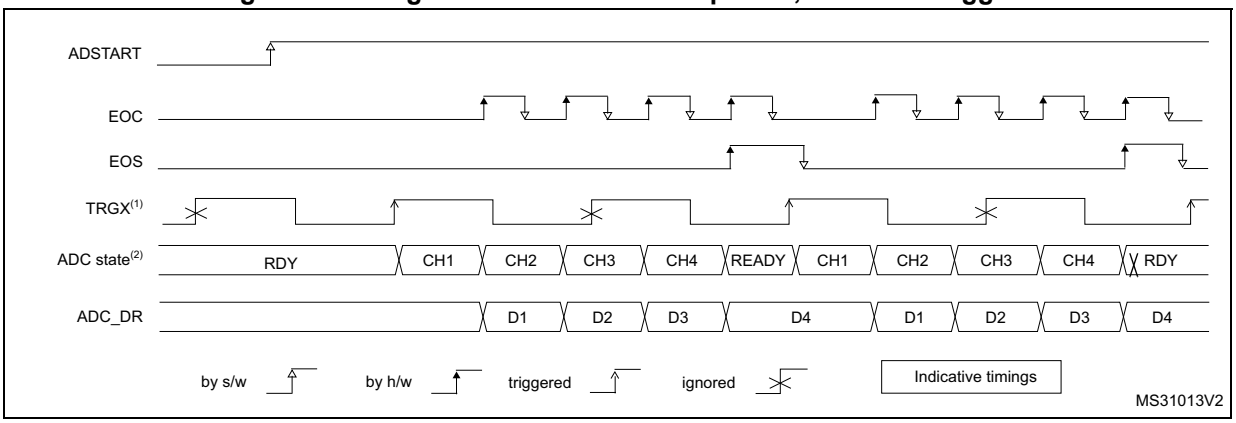
1. EXTEN = 0x0, CONT = 0
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 260. Continuous conversion of a sequence, software trigger



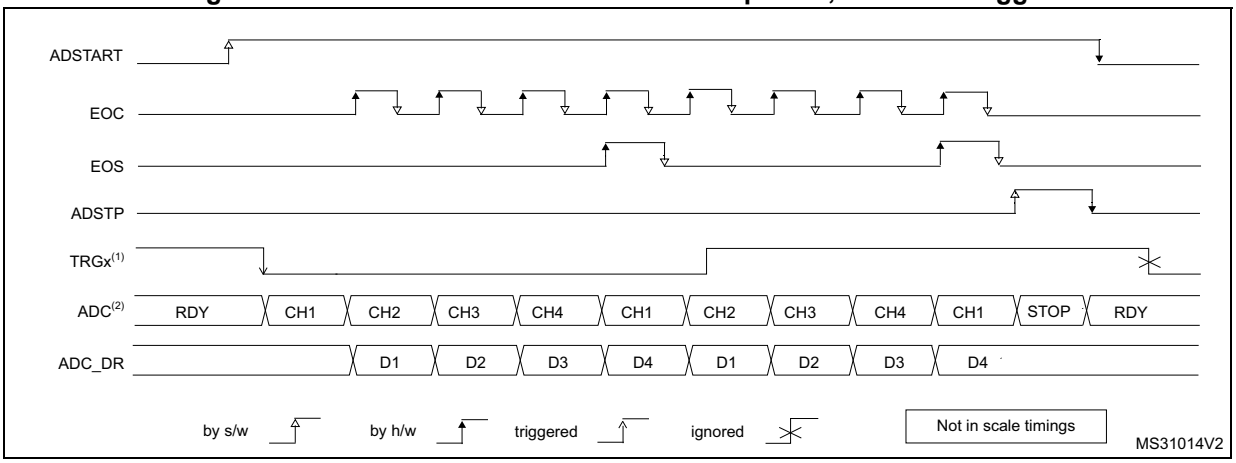
1. EXTEN = 0x0, CONT = 1
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 261. Single conversions of a sequence, hardware trigger



1. TRGx (over-frequency) is selected as trigger source, EXTEN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 262. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

29.4.25 Data management

Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADC_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 263*, *Figure 264*, *Figure 265* and *Figure 266*.

Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. In that case, the data are aligned on a byte basis as shown in *Figure 265* and *Figure 266*.

Note: Left-alignment is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the ALIGN bit value is ignored and ADC only provides right-aligned data.

Offset

An offset y (y = 1,2,3,4) can be applied to a channel by setting the bit OFFSETy_EN = 1 into ADC_OFRy register. The channel to which the offset is applied is programmed into the bits OFFSETy_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSETy[11:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

Note: Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy_EN bit in ADC_OFRy register is ignored (considered as reset).

Table 332 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 330. Offset computation versus data resolution

Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed 12-bit data	-
01: 10-bit	DATA[11:2],00	OFFSET[11:0]	Signed 10-bit data	The user must configure OFFSET[1:0] to “00”

Table 330. Offset computation versus data resolution (continued)

Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
10: 8-bit	DATA[11:4],0000	OFFSET[11:0]	Signed 8-bit data	The user must configure OFFSET[3:0] to “0000”
11: 6-bit	DATA[11:6],000000	OFFSET[11:0]	Signed 6-bit data	The user must configure OFFSET[5:0] to “000000”

When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y = 1,2,3,4) corresponding to the channel “i”:

- If one of the offsets is enabled (bit OFFSETy_EN = 1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

Figure 263, *Figure 264*, *Figure 265* and *Figure 266* show alignments for signed and unsigned data.

Figure 263. Right alignment (offset disabled, unsigned value)

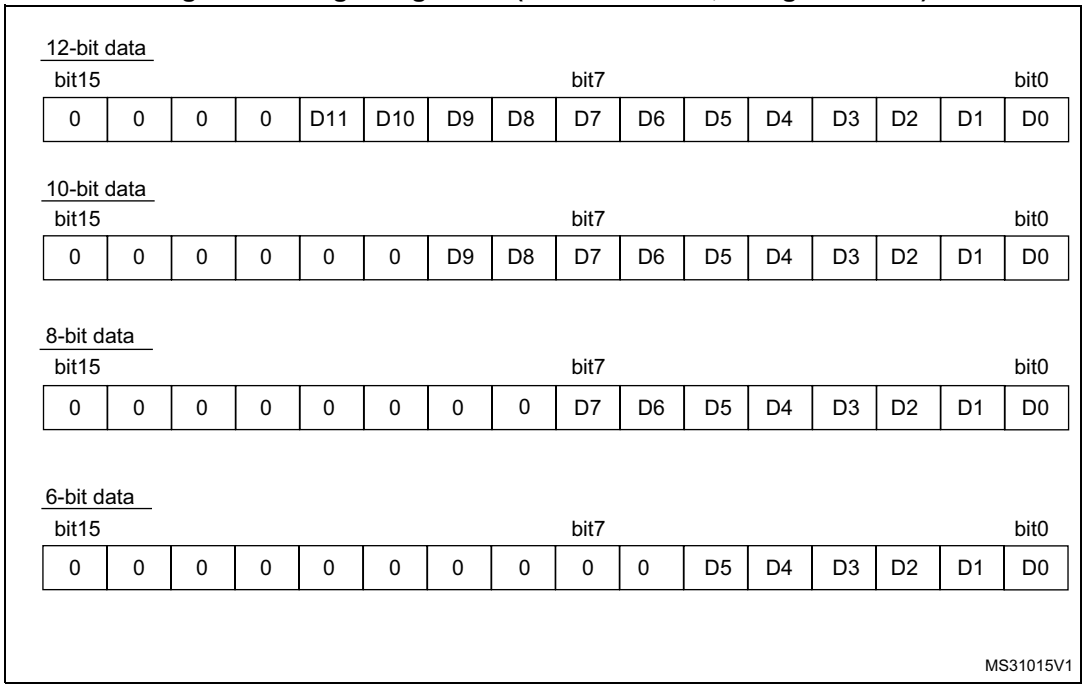


Figure 264. Right alignment (offset enabled, signed value)

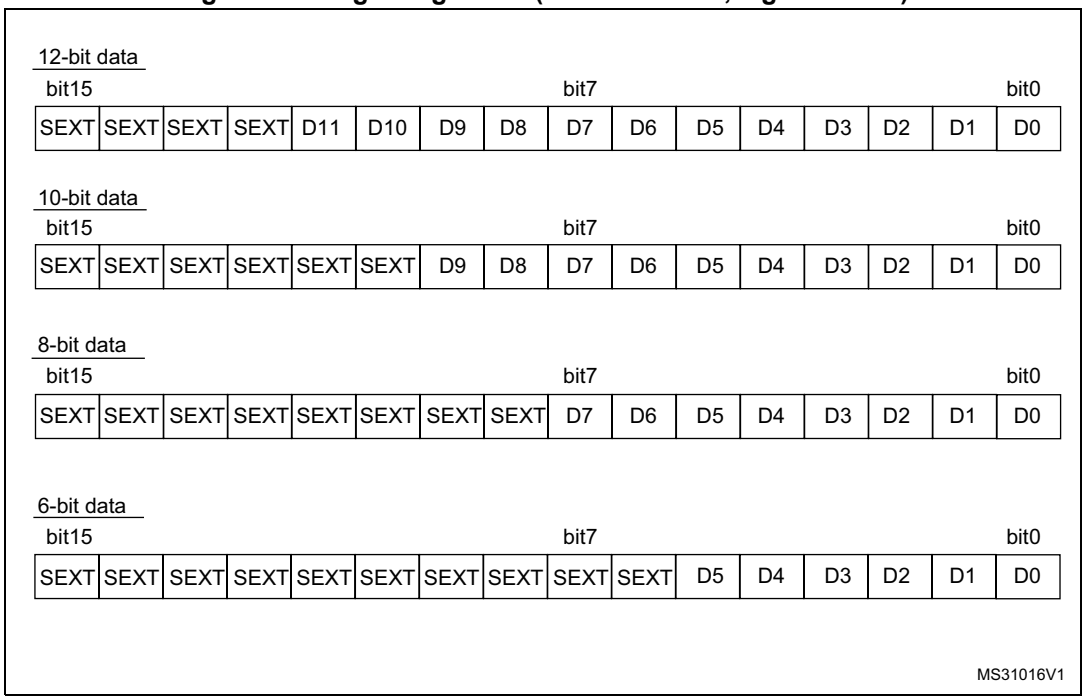


Figure 265. Left alignment (offset disabled, unsigned value)

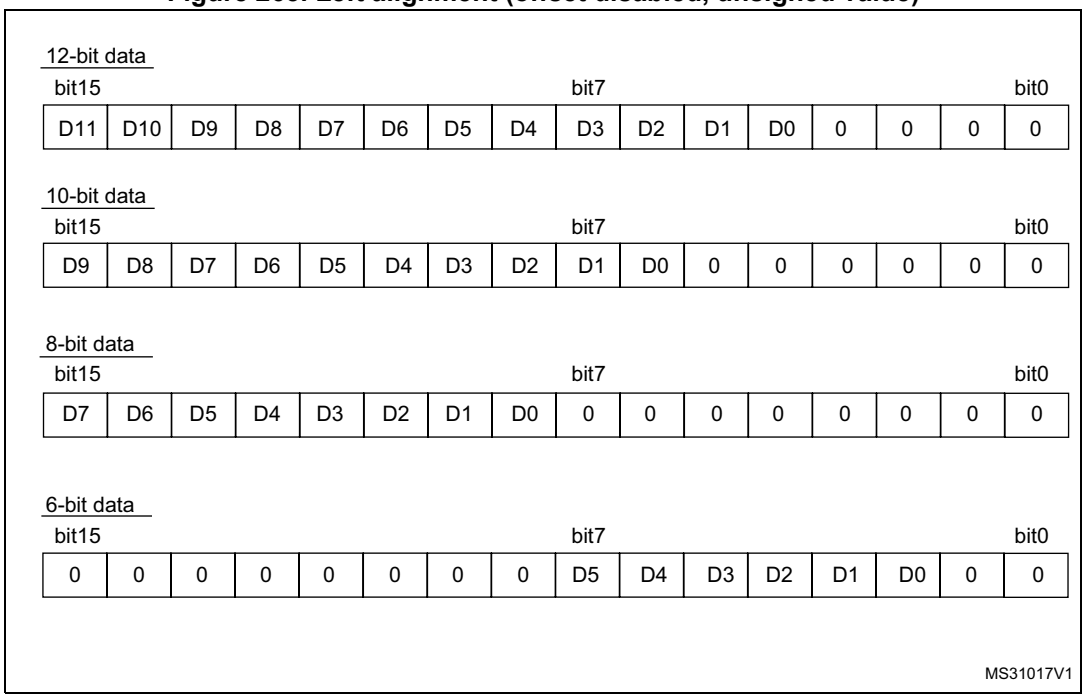
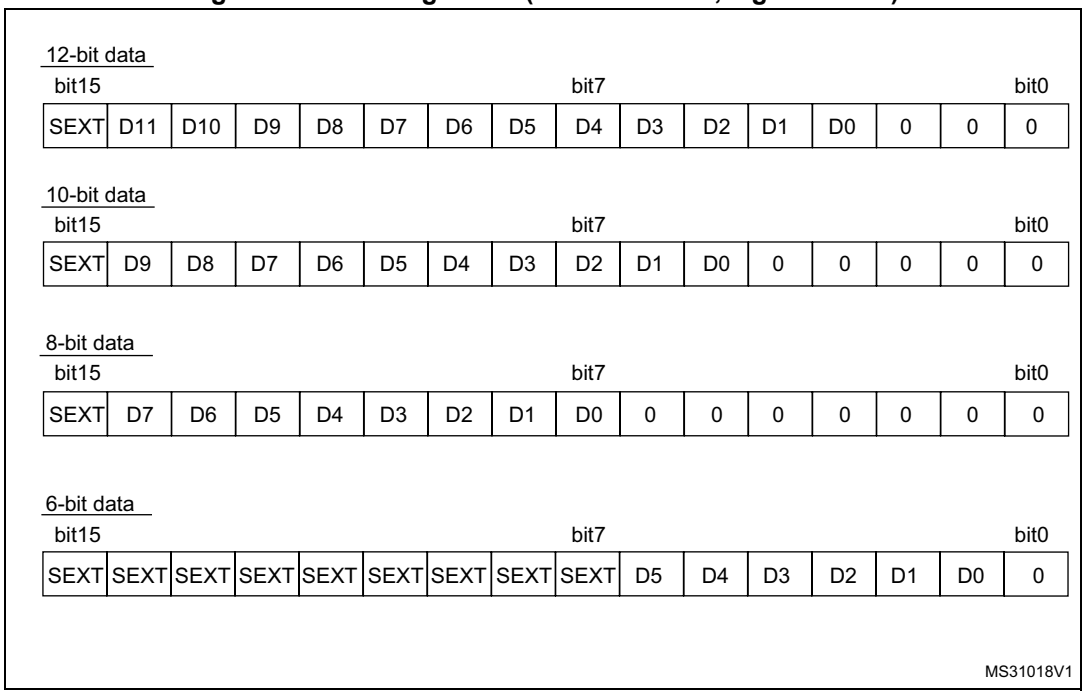


Figure 266. Left alignment (offset enabled, signed value)



Offset compensation

When SATEN bit is set in ADC_OFRy register during offset operation, data are unsigned. All the offset data saturate at 0x000 (in 12-bit mode). When OFFSETPOS bit is set, the offset direction is positive and the data saturate at 0xFFF (in 12-bit mode). In 8-bit mode, data saturate at 0x00 and 0xFF, respectively.

The analog watchdog comparison is performed before the offset compensation.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies when the regular converted data has not been read (by the CPU or the DMA) before ADC_DR FIFO (three stages) is overflowed.

The OVR flag is set when a new conversion completes while ADC_DR register FIFO was full. An interrupt is generated if OVRIE bit is set to 1.

When an overrun condition occurs, ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP to 1. Since ADC_DR FIFO features three stages, up to three data are stored in the FIFO.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0: The overrun event preserves the data register from being overwritten: the old data is maintained up to ADC_DR FIFO depth (three stages) and the new conversion is discarded and lost. In this mode, ADC_DR FIFO is enabled. If the FIFO is full, any further conversion is performed but the resulting data is also discarded. EOC is

cleared by reading ADC_DR register. However, the FIFO can still contain previously converted data.

- OVRMOD = 1: The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversion is performed normally and the ADC_DR register always contains the latest converted data.

Figure 267. Example of overrun (OVRMOD = 0)

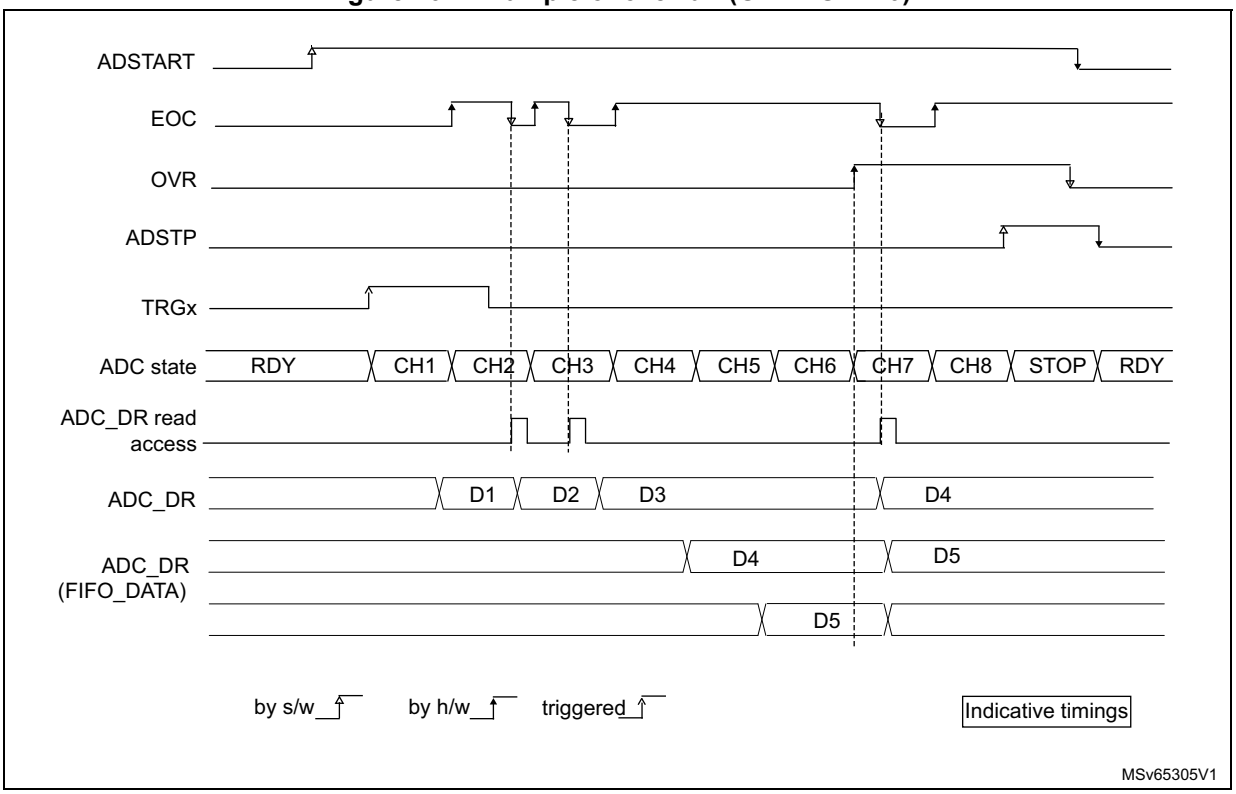
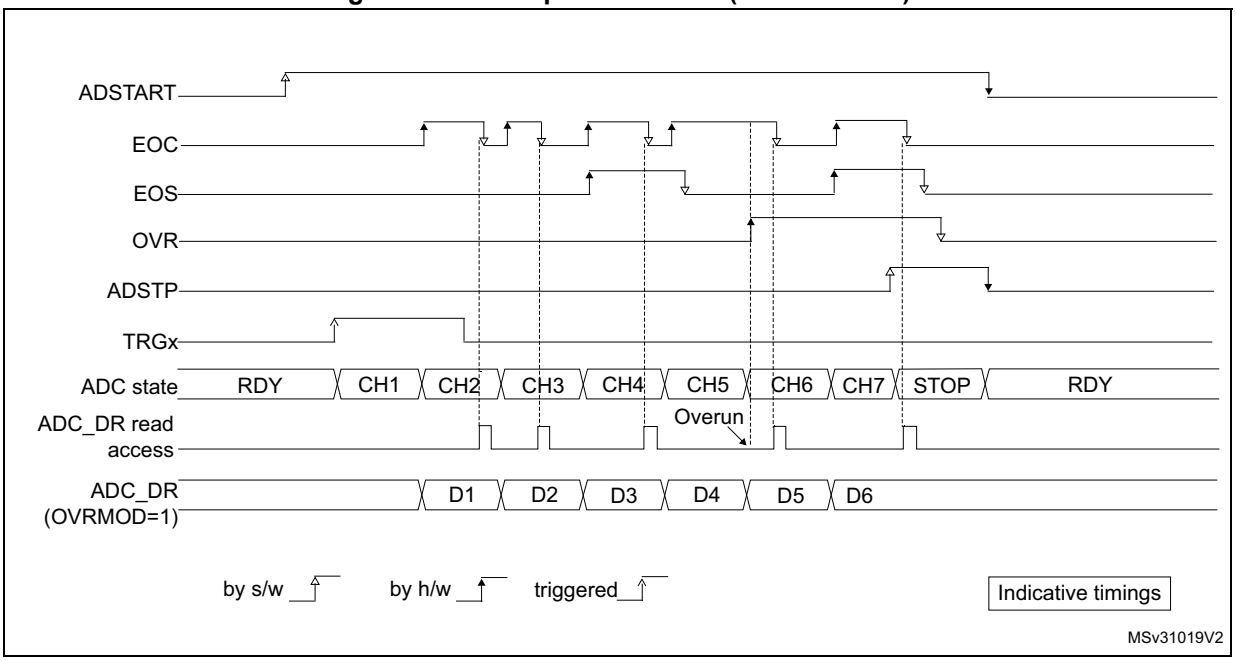


Figure 268. Example of overrun (OVRMOD = 1)



Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag must be ignored by the software. An overrun event does not prevent ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if an overrun occurs (OVR = 1) because the DMA could not serve the DMA transfer request in time, ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to *ADC overrun (OVR, OVRMOD)*).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG of the ADC_CFGR register in single ADC mode, or with bit DMACFG of the ADC_CCR register in dual ADC mode:

- DMA one shot mode (DMACFG = 0).
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMACFG = 1)
  This mode is suitable when programming the DMA in circular mode.

DMA one shot mode (DMACFG = 0)

In this mode, ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

DMA circular mode (DMACFG = 1)

In this mode, ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

29.4.26 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY = 1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see *Figure 269*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 270*).

This is a way to automatically adapt the speed of ADC to the speed of the system which reads the data.

The delay is inserted after each regular conversion (whatever DISCEN = 0 or 1) and after each sequence of injected conversions (whatever JDISCEN = 0 or 1).

Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

Note: This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to restart a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see *Figure 270*).
- Once the injected sequence is complete, ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see *Figure 272*).

The behavior is slightly different in auto-injected mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 273*).

To stop a conversion in Continuous auto-injection mode combined with autodelay mode (JAUTO = 1, CONT = 1 and AUTDLY = 1), follow the following procedure:

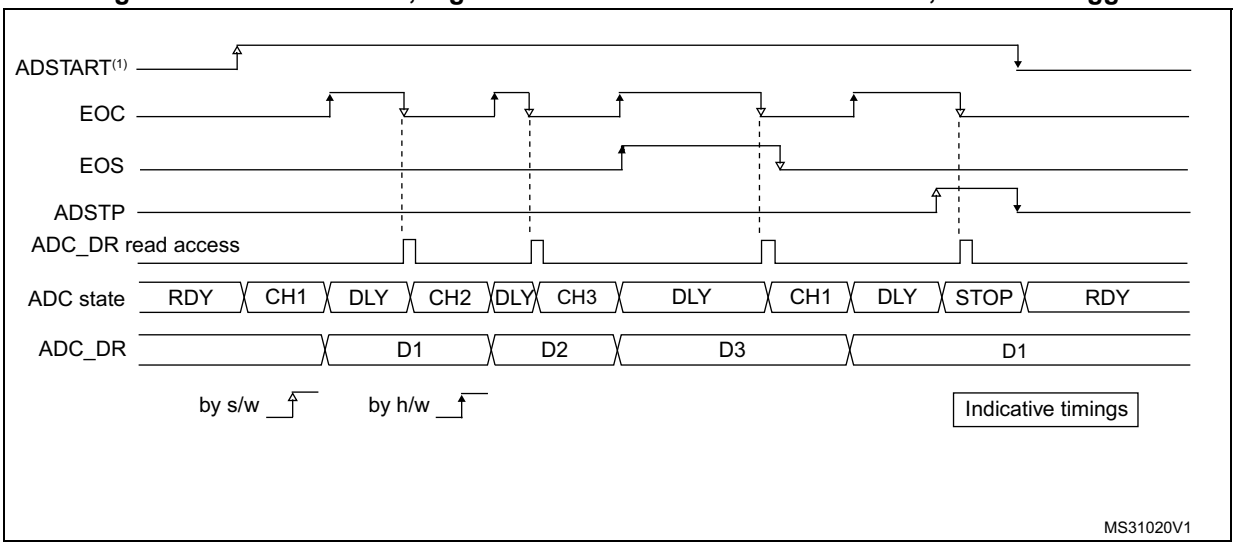
1. Wait until JEOS = 1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP = 1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can restart if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

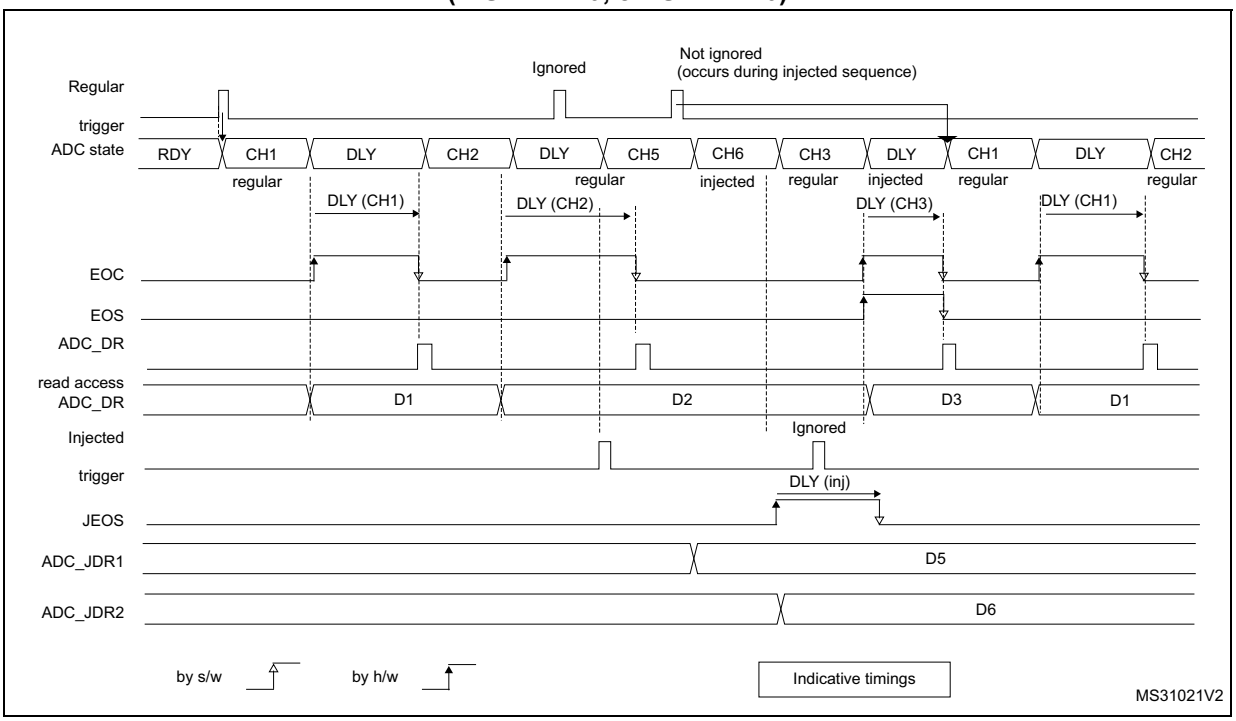
Figure 269. AUTODLY = 1, regular conversion in Continuous mode, software trigger



1. AUTDLY = 1

Note: Regular configuration: EXTEN=0x0 (SW trigger), CONT = 1, CHANNELS = 1,2,3
Injected configuration DISABLED

Figure 270. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)

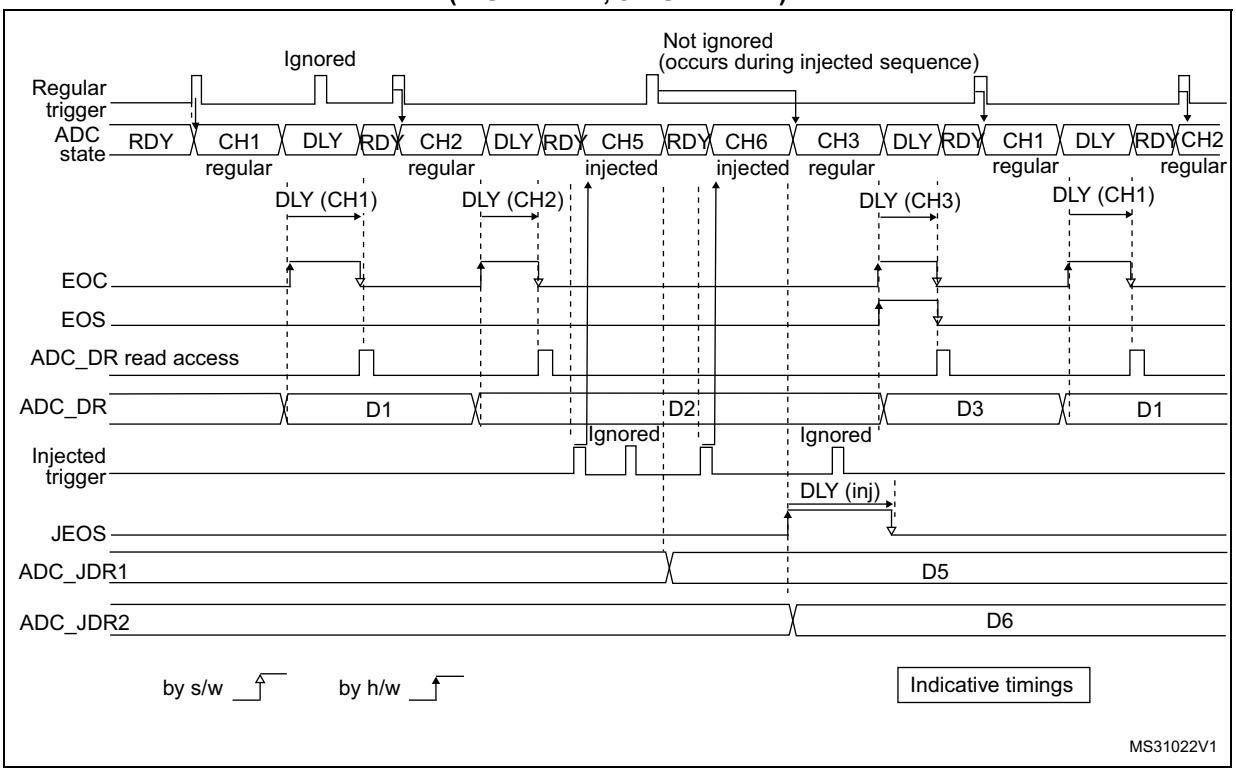


Note: AUTDLY = 1

Regular configuration: EXTEN=0x1 (HW trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3

Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

Figure 271. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)

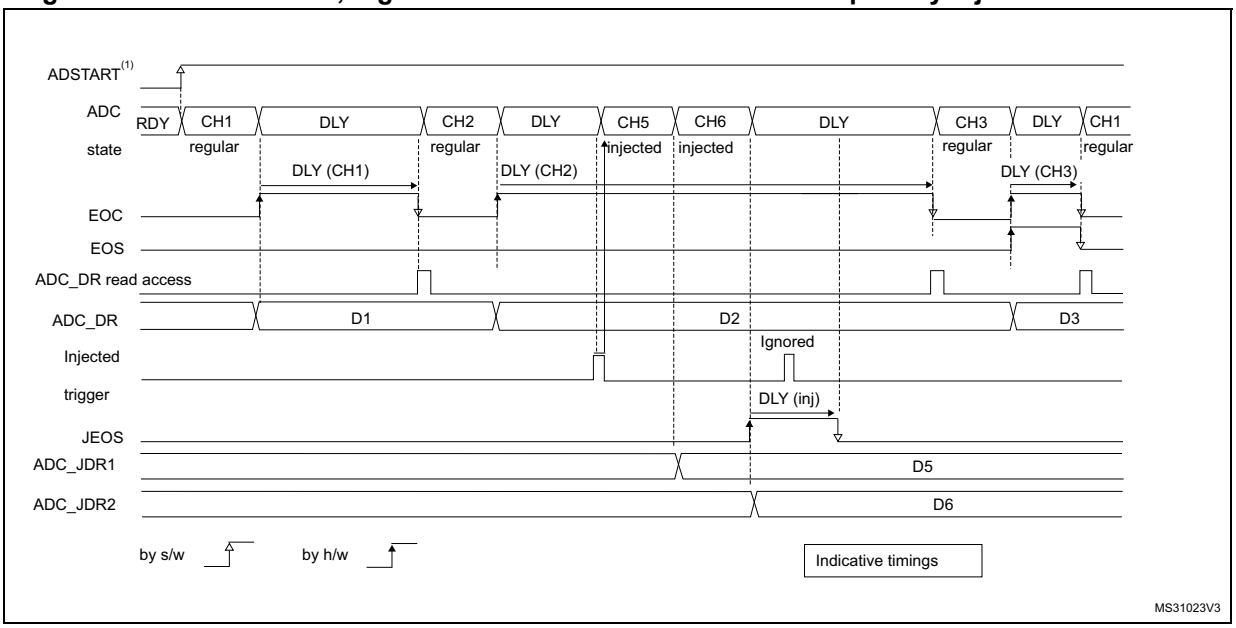


Note: AUTDLY = 1

Regular configuration: EXTEN = 0x1 (HW trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.

Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 1, CHANNELS = 5,6

Figure 272. AUTODLY = 1, regular continuous conversions interrupted by injected conversions

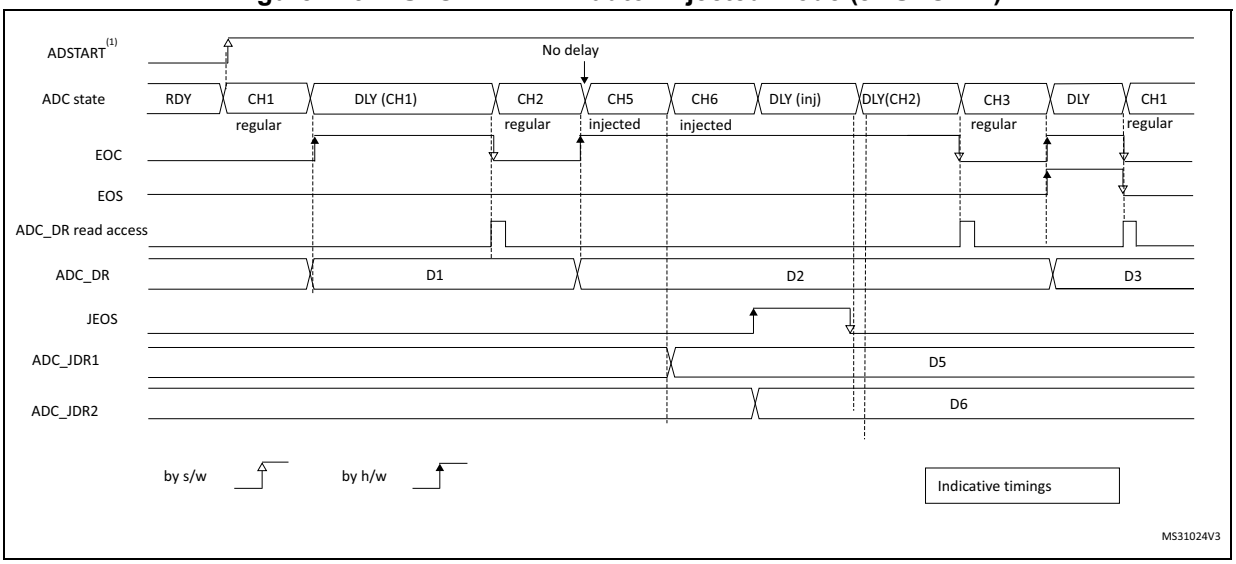


1. AUTDLY = 1

Note: Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3

Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

Figure 273. AUTODLY = 1 in auto- injected mode (JAUTO = 1)



1. AUTDLY = 1

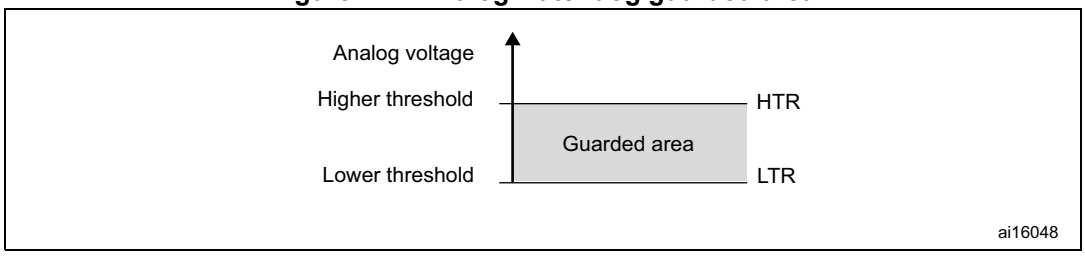
Note: Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2

Injected configuration: JAUTO = 1, CHANNELS = 5,6

29.4.27 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 274. Analog watchdog guarded area



AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADC_IER register (x = 1,2,3).

AWDx (x = 1,2,3) flag is cleared by software by writing 1 to it.

ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels remain within a configured voltage range (window).

Table 331 shows how the ADC_CFGR registers must be configured to enable the analog watchdog on one or more channels.

Table 331. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed in bits HT1[11:0] and LT1[11:0] of the ADC_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits

(according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned) before the offset compensation stage.

Table 332 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 332. Analog watchdog 1 comparison

Resolution(bit RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
00: 12-bit	DATA[11:0]	LT1[11:0] and HT1[11:0]	-
01: 10-bit	DATA[11:2],00	LT1[11:0] and HT1[11:0]	User must configure LT1[1:0] and HT1[1:0] to 00
10: 8-bit	DATA[11:4],0000	LT1[11:0] and HT1[11:0]	User must configure LT1[3:0] and HT1[3:0] to 0000
11: 6-bit	DATA[11:6],000000	LT1[11:0] and HT1[11:0]	User must configure LT1[5:0] and HT1[5:0] to 000000

Analog watchdog filter for watchdog 1

When an ADC is configured with only one input channel (selecting several channels in Scan mode not allowed), a valid ADC conversion data interval can be configured through the ADC_TR1 register:

- When converted data belong to the interval defined in ADC_TR1, a DMA request is generated.
- Otherwise, no DMA request is issued. RDATA register is updated at each conversion. If data is out-of-range a number of times higher than the value specified in AWDFILT bit of ADC_TR1, the AWDx flag is set and the corresponding interrupt is issued.

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDxCH[18:0] (x = 2,3).

The corresponding watchdog is enabled when any bit of AWDxCH[18:0] (x = 2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. *Table 333* describes how the comparison is performed for all the possible resolutions.

Table 333. Analog watchdog 2 and 3 comparison

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
00: 12-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:0] are not relevant for the comparison
01: 10-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:2] are not relevant for the comparison

Table 333. Analog watchdog 2 and 3 comparison (continued)

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
10: 8-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	-
11: 6-bit	DATA[11:6],00	LTx[7:0] and HTx[7:0]	User must configure LTx[1:0] and HTx[1:0] to 00

ADCy_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y = ADC number, x = watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCy_AWDx_OUT is also reset when disabling ADC (when setting ADDIS = 1). Note that stopping regular or injected conversions (setting ADSTP = 1 or JADSTP = 1) has no influence on the generation of ADCy_AWDx_OUT.

Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 275. ADCy_AWDx_OUT signal generation (on all regular channels)

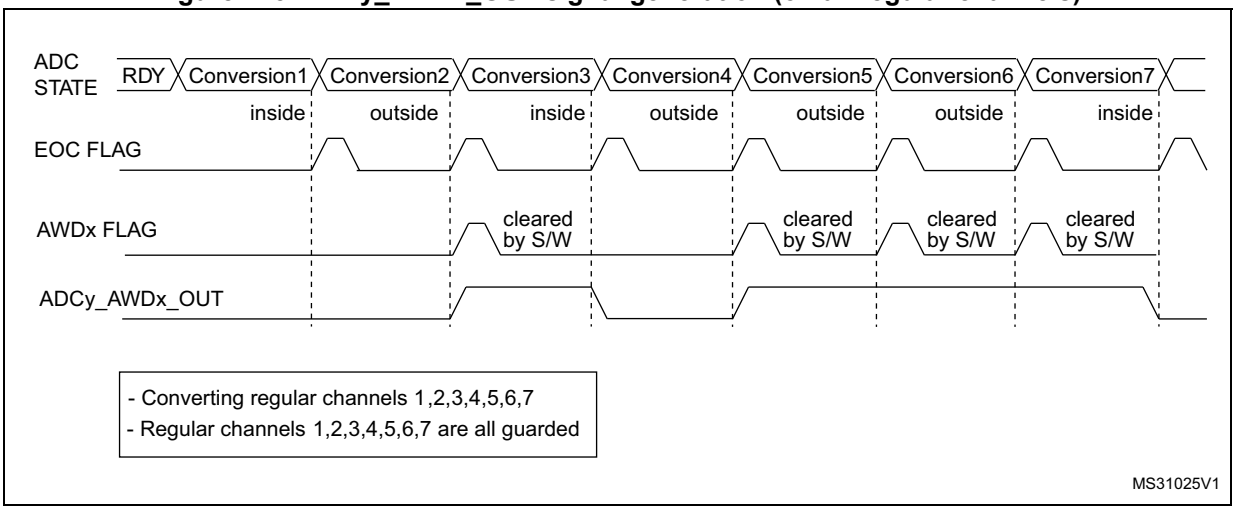


Figure 276. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software)

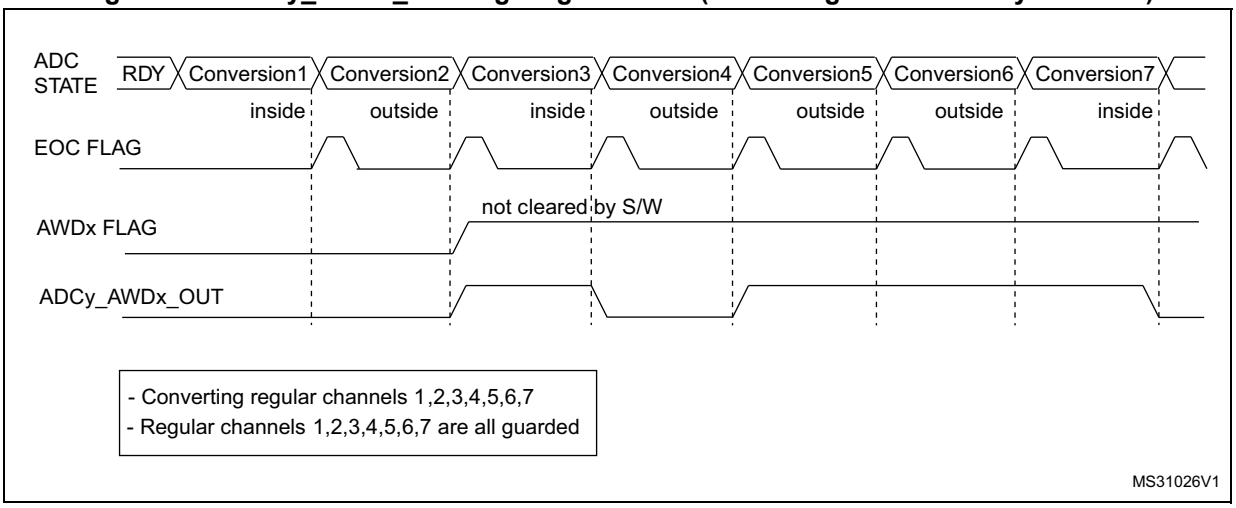


Figure 277. ADCy_AWDx_OUT signal generation (on a single regular channel)

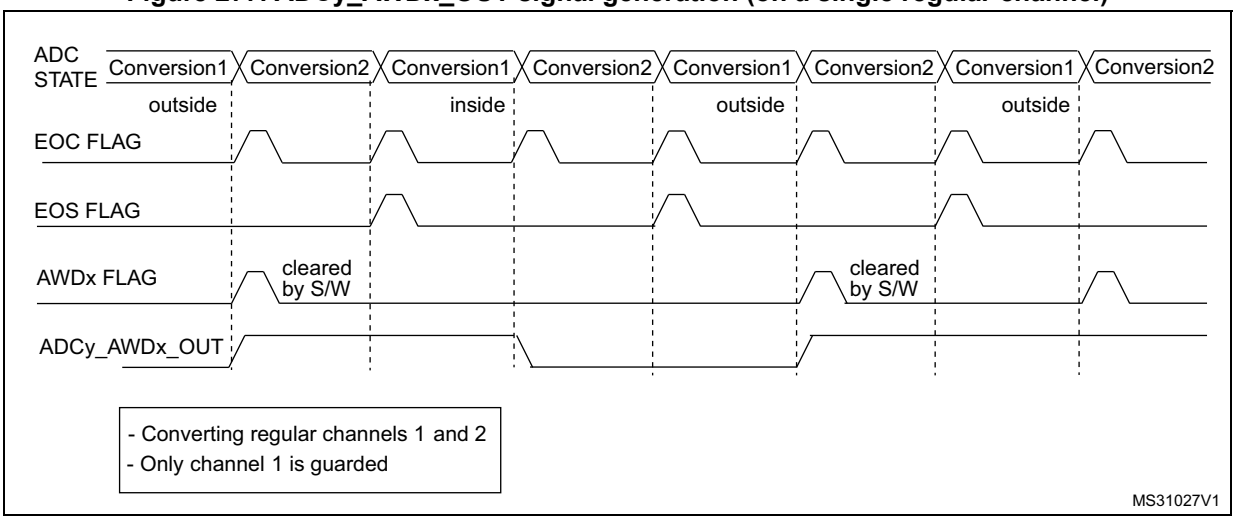
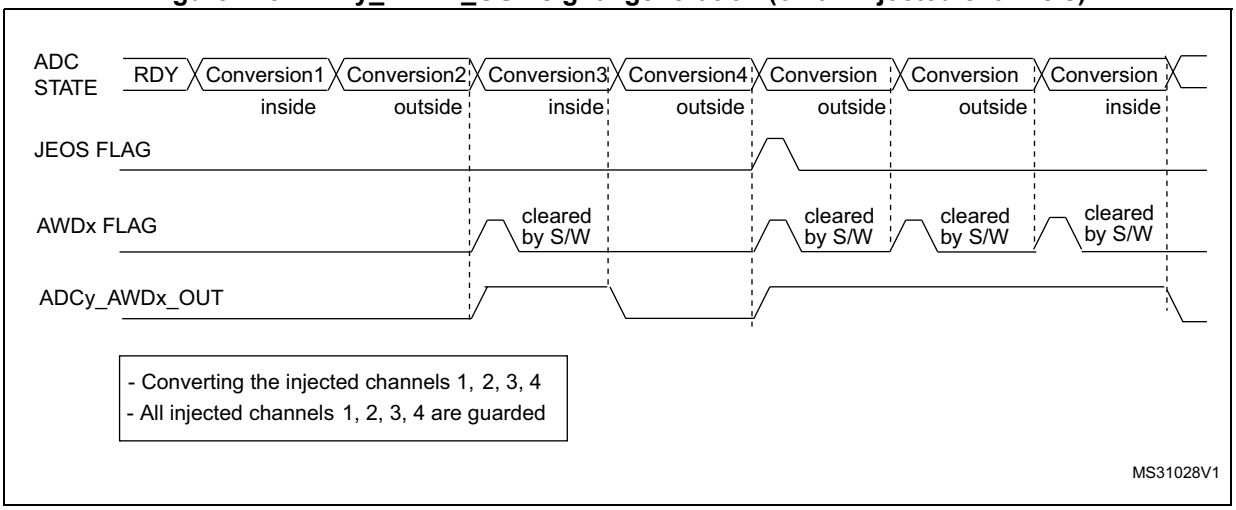


Figure 278. ADCy_AWDx_OUT signal generation (on all injected channels)



Analog watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of ADC internal state). If LTx[11:0] and HTx[11:0] are updated during ADC conversion of the ADC guarded channel, the watchdog function is masked for this conversion. This masking is removed at the next start of conversion, resulting in an analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new interval, no interrupt and AWDx_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion which started after the threshold update. If AWD_xOUT is already asserted, programming the new thresholds does not deassert the AWDx_OUT signal.

Analog watchdog with offset compensation

When the offset compensation is enabled, the analog watchdog compares the threshold before the data compensation.

29.4.28 Oversampler

The oversampling unit performs data pre-processing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVSR[2:0] bits in the ADC_CFGR2 register, and can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to the 16 least significant bits, rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

Note: If the intermediary result after the shifting exceeds 16-bit, the result is truncated as it is, without saturation.

Figure 279. 20-bit to 16-bit result truncation

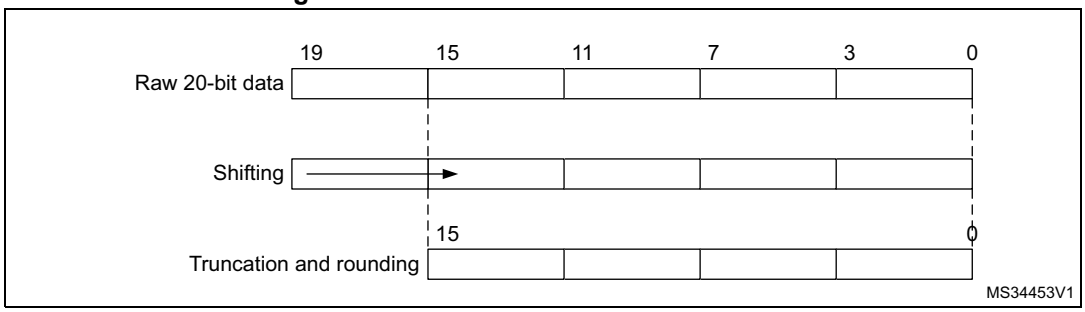


Figure 280 gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 280. Numerical example with 5-bit shift and rounding

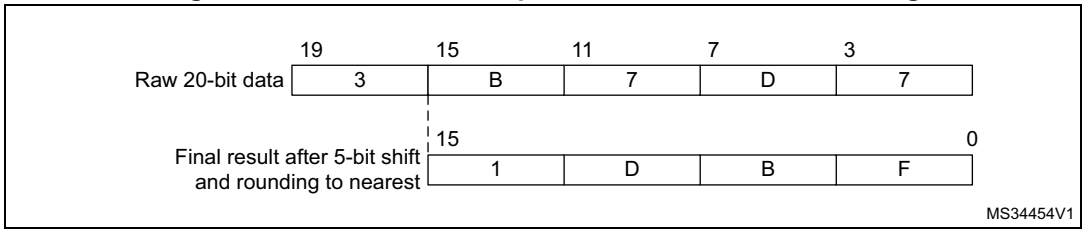


Table 334 gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 334. Maximum output results versus N and M (gray cells indicate truncation)

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS = 0111	8-bit shift OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N

conversions, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase
- The end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after N x sequence length conversions total)

ADC operating modes supported when oversampling (single ADC mode)

In oversampling mode, most of ADC operating modes are maintained:

- Single or continuous conversion modes
- ADC conversions start either by software or with triggers
- ADC stops during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

Note: The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR1 is ignored and the data are always provided right-aligned.

Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy_EN bit in ADC_OFRy register is ignored (considered as reset).

Analog watchdog

The analog watchdog functionality is maintained (AWDSGL and AWDEN bits), with the following difference:

- The RES[1:0] bits are ignored, comparison is always done on using the full 12-bit values HT[11:0] and LT[11:0]
- the comparison is performed on the most significant 12-bit of the 16-bit oversampled results ADC_DR[15:4]

Note: Care must be taken when using high shifting values, this reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HT[0:7] / LT[[0:7], and HT[11:8] / LT[11:8] must be kept reset.

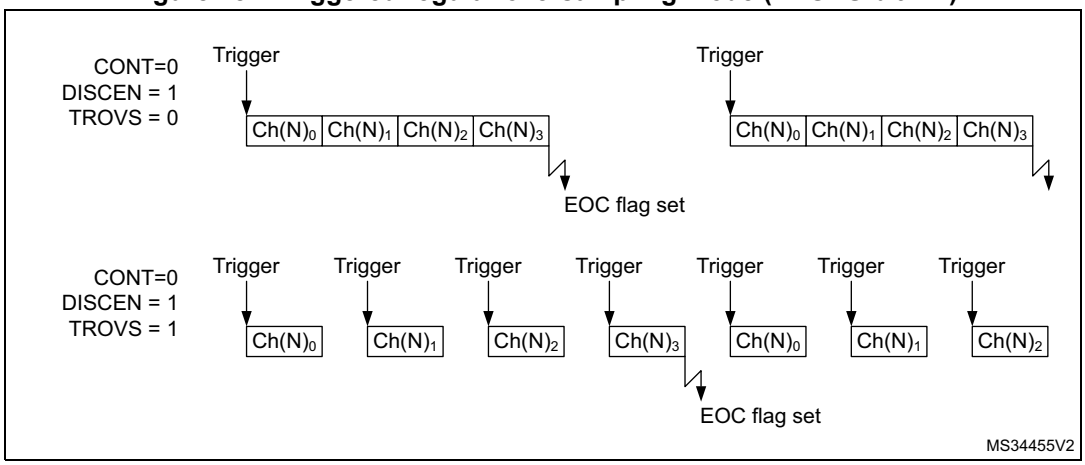
Triggered mode

The averaging can also be used for basic filtering purpose. Although it is not a very powerful filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific Discontinuous mode can be enabled with TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

The *Figure 281* below shows how conversions are started in response to triggers during Discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 281. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behaviors for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

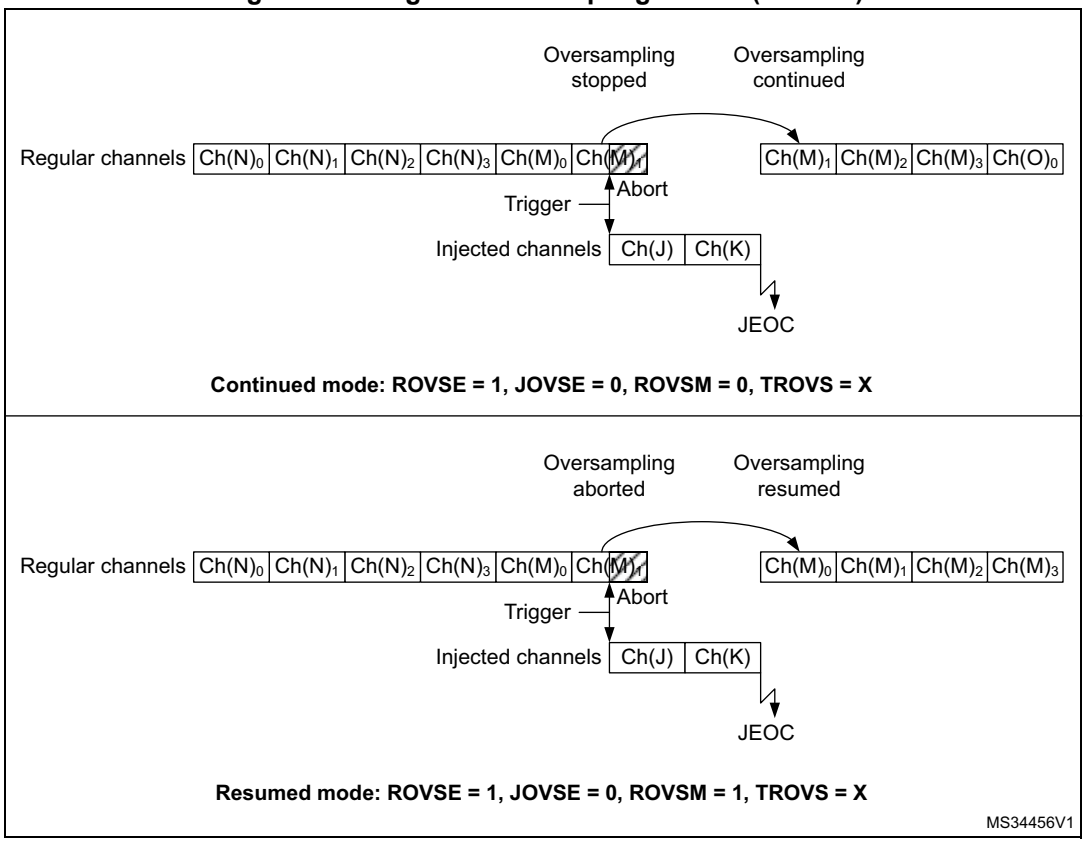
Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing that at least one regular conversion can be completed between triggers);
- In resumed mode, the accumulation restarts from 0 (previous conversions results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have an injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be completed and the regular sequencer is blocked.

The *Figure 282* gives examples for a 4x oversampling ratio.

Figure 282. Regular oversampling modes (4x ratio)



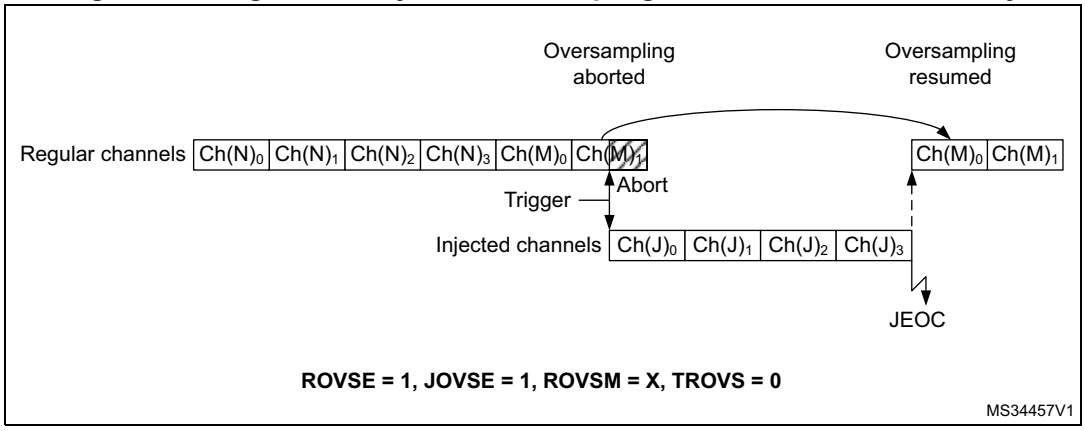
Oversampling Injected channels only

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented in *Figure 283* below.

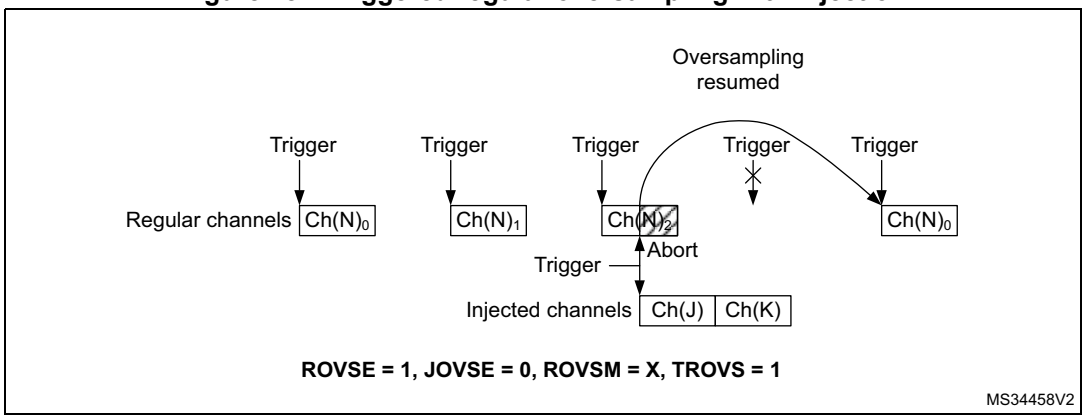
Figure 283. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented in *Figure 284* below.

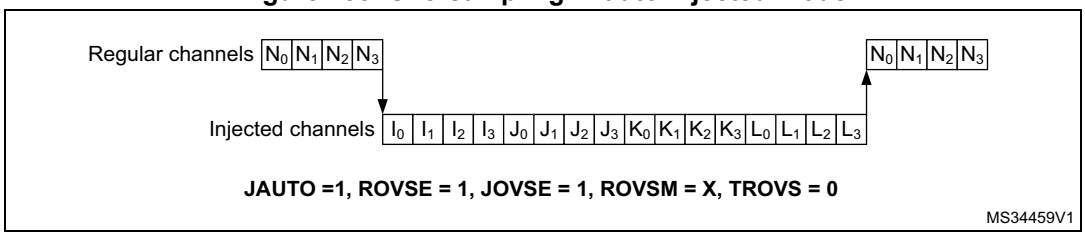
Figure 284. Triggered regular oversampling with injection



Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The *Figure 285* below shows how the conversions are sequenced.

Figure 285. Oversampling in auto-injected mode



It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, ADC must be configured as follows: JAUTO = 1, DISCEN = 0, JDISCEN = 0, ROVSE = 1, JOVSE = 1 and TROVSE = 1.

Dual ADC modes supported when oversampling

It is possible to have oversampling enabled when working in dual ADC configuration, for the injected simultaneous mode and regular simultaneous mode. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (ROVSE = 1 or JOVSE = 1).

Combined modes summary

The *Table 335* below summarizes all combinations, including modes not supported.

Table 335. Oversampler operating modes summary

Regular Oversampling ROVSE	Injected Oversampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered Regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

29.4.29 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see *Figure 286*).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by ADCx master to ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCx_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode
- Injected simultaneous mode + Interleaved mode

In dual ADC mode (when bits DUAL[4:0] in ADCx_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JQM, JAUTO of the ADC_CFGR register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

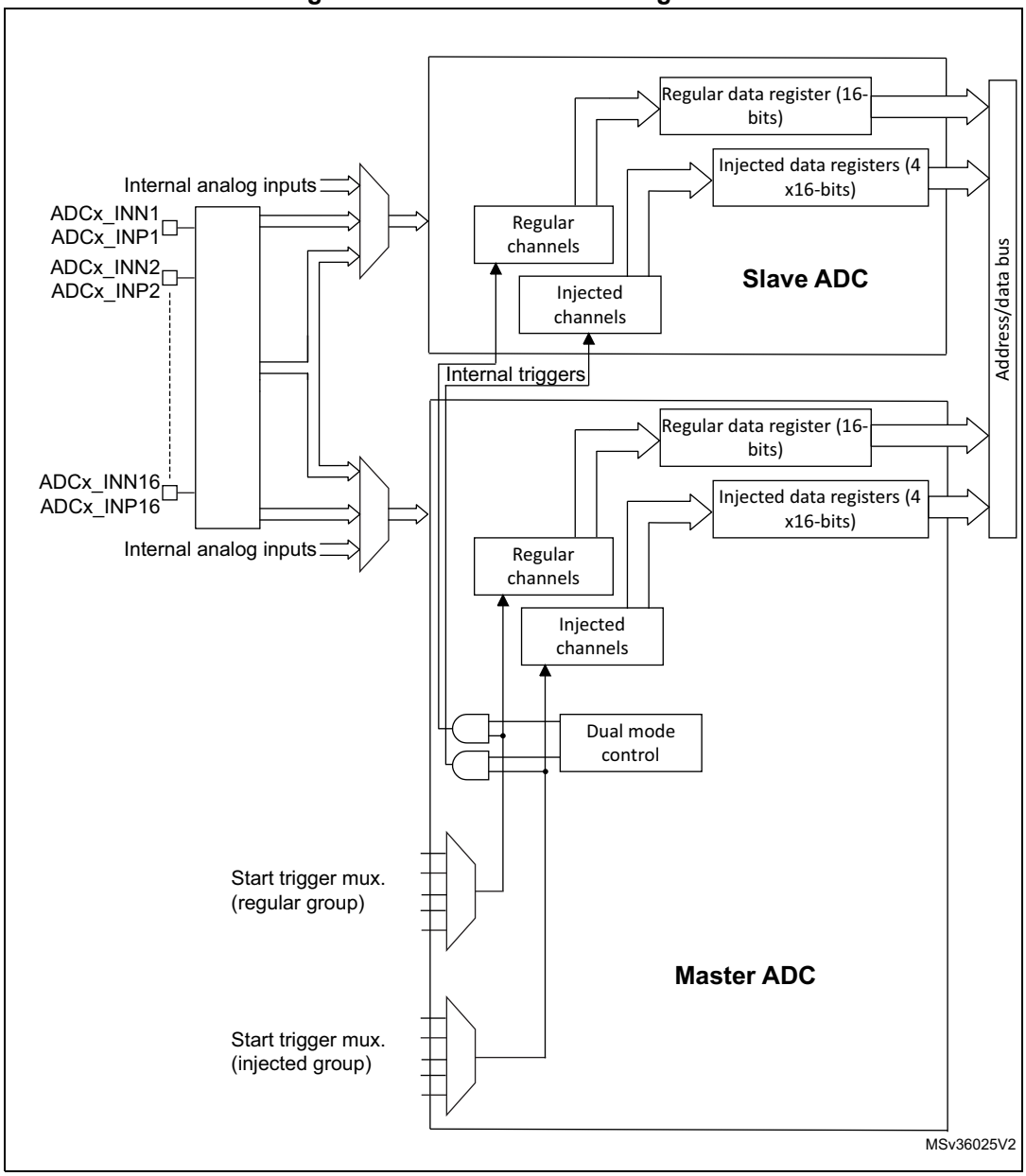
To start a conversion in dual mode, the user must program the bits EXTEN, EXTSEL, JEXTEN, JEXTSEL of the master ADC only, to configure a software or hardware trigger, and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don’t care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCx_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADCx_CSR).

Figure 286. Dual ADC block diagram(1)



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.

Note: The ADC common data register (ADCx_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

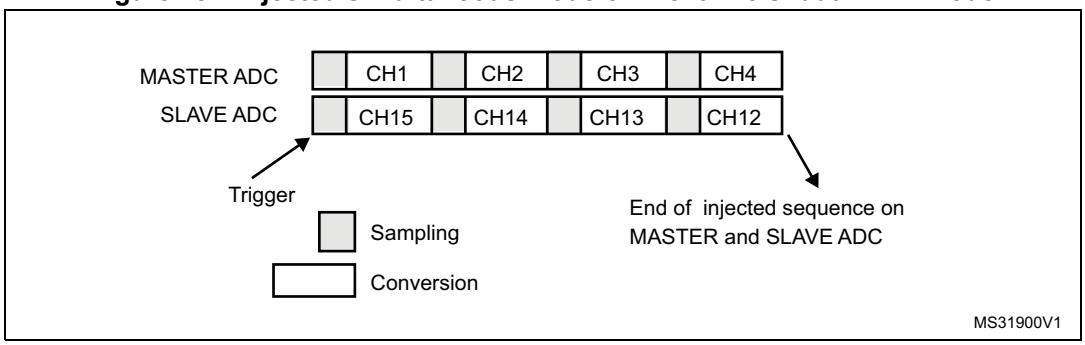
Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longest of the 2 sequences. Otherwise, ADC with the shortest sequence may restart while ADC with the longest sequence is completing the previous conversions.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in *Figure 287*), it is possible for the software to enable only one of the two JEOS interrupts (such as master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 287. Injected simultaneous mode on 4 channels: dual ADC mode



If JDISCEN = 1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the

ongoing regular sequence and the associated delay phases are ignored.
There is the same behavior for regular sequences occurring on the slave ADC.

Regular simultaneous mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) aborts the current simultaneous conversions, which are restarted once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longest conversion time of the 2 sequences. Otherwise, ADC with the shortest sequence may restart while ADC with the longest sequence is completing the previous conversions.

Software is notified by interrupts when it can read the data:

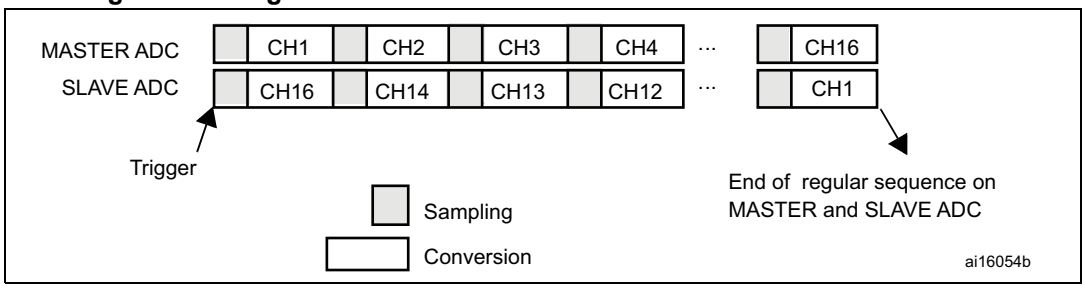
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in *Figure 288*), it is possible for the software to enable only one of the two EOC interrupts (ex: master EOC) and read both converted data from the Common Data register (ADCx_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using two DMA channels (one for the master and one for the slave). In this case bits MDMA[1:0] must be kept cleared.
  - Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  - Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Using MDMA mode, which leaves one DMA channel free for other uses:
  - Configure MDMA[1:0] = 0b10 or 0b11 (depending on resolution).
  - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR)
  - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
  - Both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Note: In MDMA mode (MDMA[1:0] = 0b10 or 0b11), the user must program the same number of conversions in the master's sequence as in the slave's sequence. Otherwise, the remaining conversions do not generate a DMA request.

Figure 288. Regular simultaneous mode on 16 channels: dual ADC mode



If DISCEN = 1 then each “n” simultaneous conversion of the regular sequence requires a regular trigger event to occur (“n” is defined by DISCNUM).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, ADCx_CDR (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADCx_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits MDMA must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master’s sequence is equal to the number of conversions in the slave’s.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

Note: This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

Interleaved mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00111.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates two conversions in interleaved mode is configured in the DELAY bits in the ADCx_CCR register. This delay starts counting one half cycle after the end of the sampling phase of the master conversion. This way, an ADC cannot start a

conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Note: It is possible to enable only the EOC interrupt of the slave and read the common data register (ADCx_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts. It is recommended to use the MDMA mode.

It is also possible to have the regular data transferred by DMA. In this case, individual DMA requests on each ADC cannot be used and it is mandatory to use the MDMA mode, as follows:

- Configure MDMA[1:0] = 0b10 or 0b11 (depending on resolution).
- A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR).
- A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
- Both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Figure 289. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode

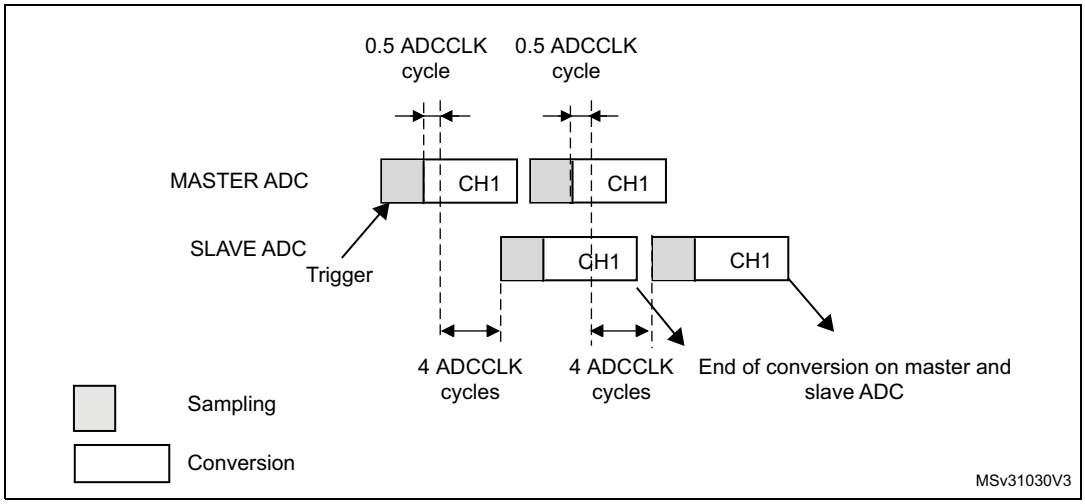
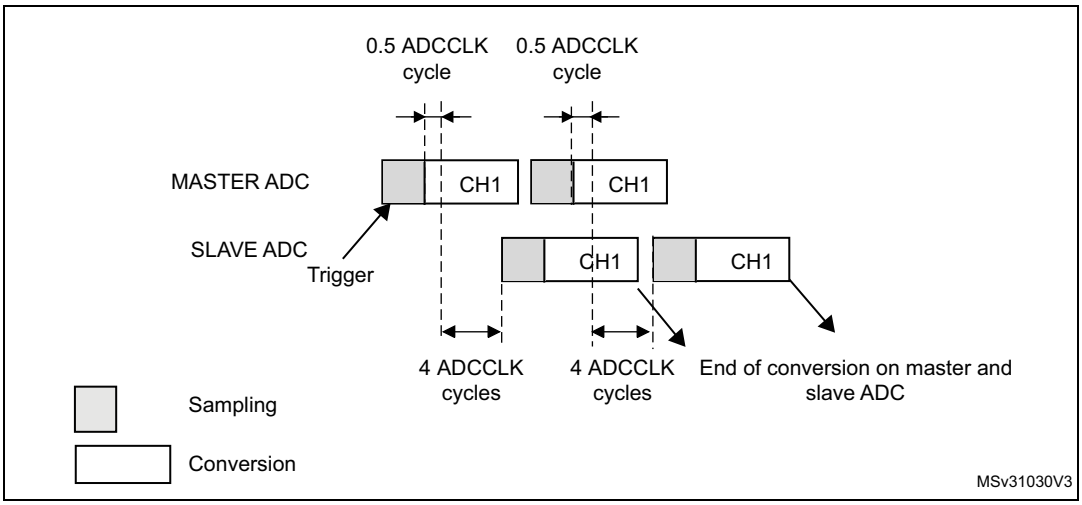


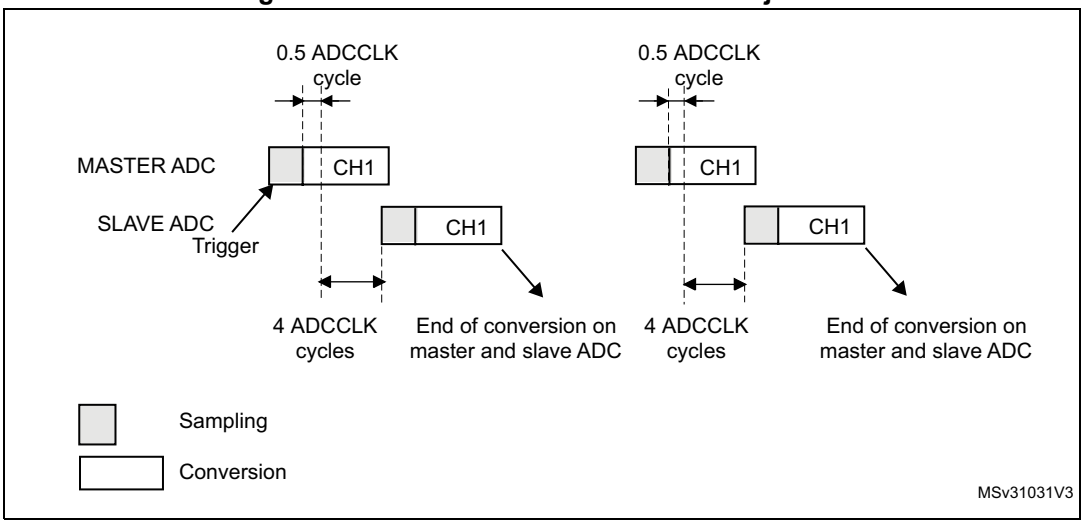
Figure 290. Interleaved mode on 1 channel in single conversion mode: dual ADC mode



If DISCEN = 1, each “n” simultaneous conversion (“n” is defined by DISCNUM) of the regular sequence requires a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is restarted from the master (see *Figure 291* below).

Figure 291. Interleaved conversion with injection



Alternate trigger mode

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN must not be 0x0.

Injected discontinuous mode disabled (JDISCEN = 0 for both ADC)

1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

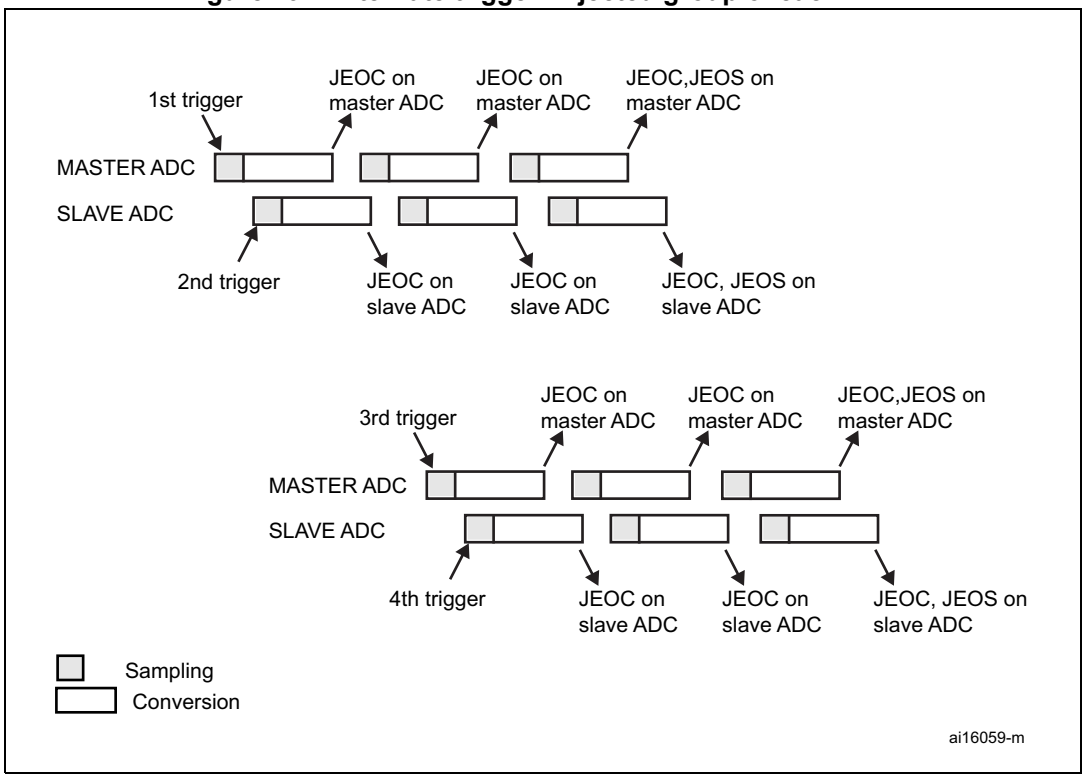
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 292. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN = 1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

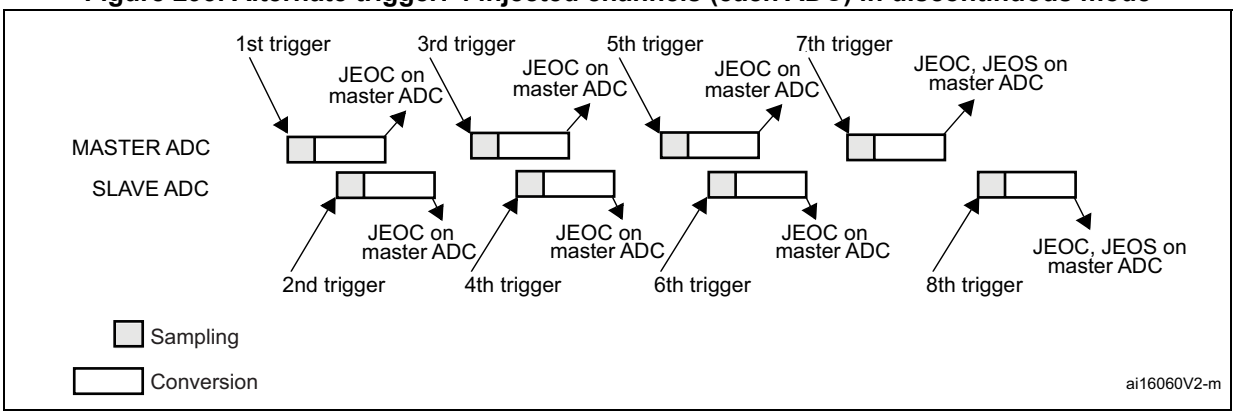
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 293. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longest conversion time of the 2 sequences. Otherwise, ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous + alternate trigger mode

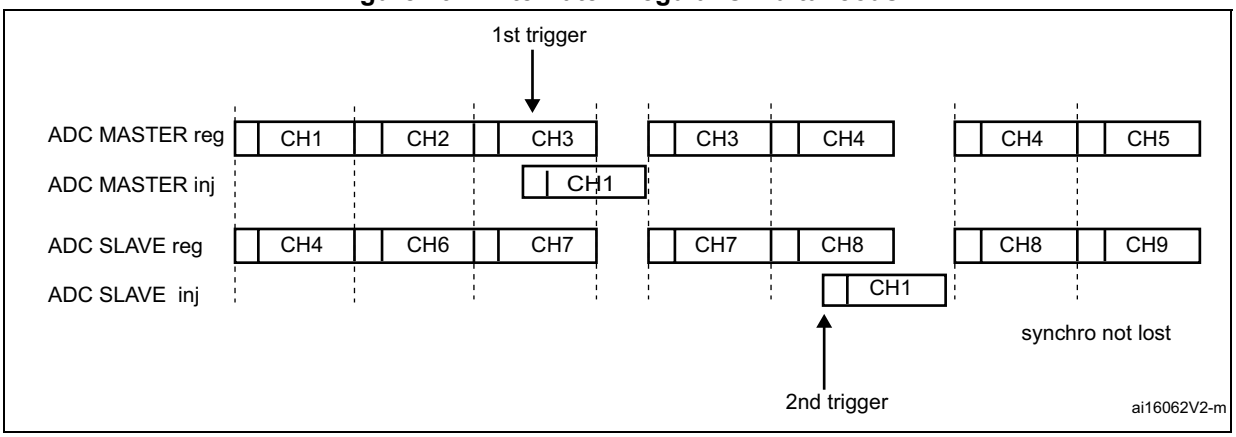
This mode is selected by programming bits DUAL[4:0] = 00010.

It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. *Figure 294* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

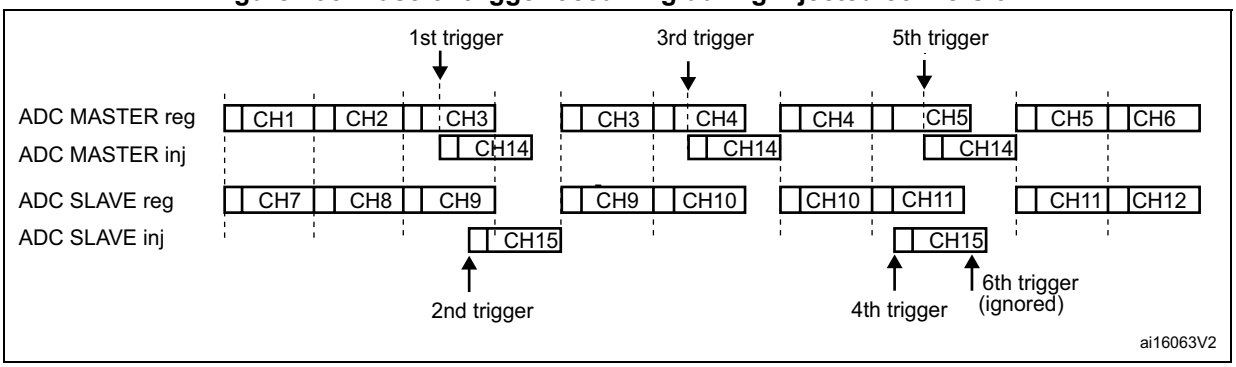
Note: In combined regular simultaneous + alternate trigger mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, ADC with the shortest sequence may restart while ADC with the longest sequence is completing the previous conversions.

Figure 294. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 295* shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

Figure 295. Case of trigger occurring during injected conversion



Combined injected simultaneous plus interleaved

This mode is selected by programming bits DUAL[4:0] = 00011

It is possible to interrupt an interleaved conversion with a simultaneous injected event.

In this case the interleaved conversion is interrupted immediately and the simultaneous injected conversion starts. At the end of the injected sequence the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is always the master's one. *Figure 296*, *Figure 297* and *Figure 298* show the behavior using an example.

Caution: In this mode, it is mandatory to use the Common Data Register to read the regular data with a single read access. On the contrary, master-slave data coherency is not guaranteed.

Figure 296. Interleaved single channel CH0 with injected sequence CH11, CH12

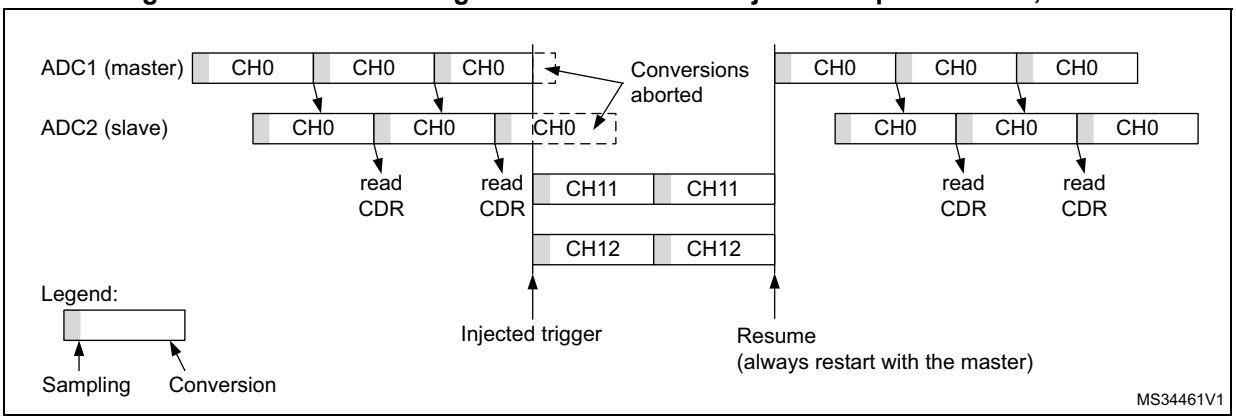


Figure 297. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first

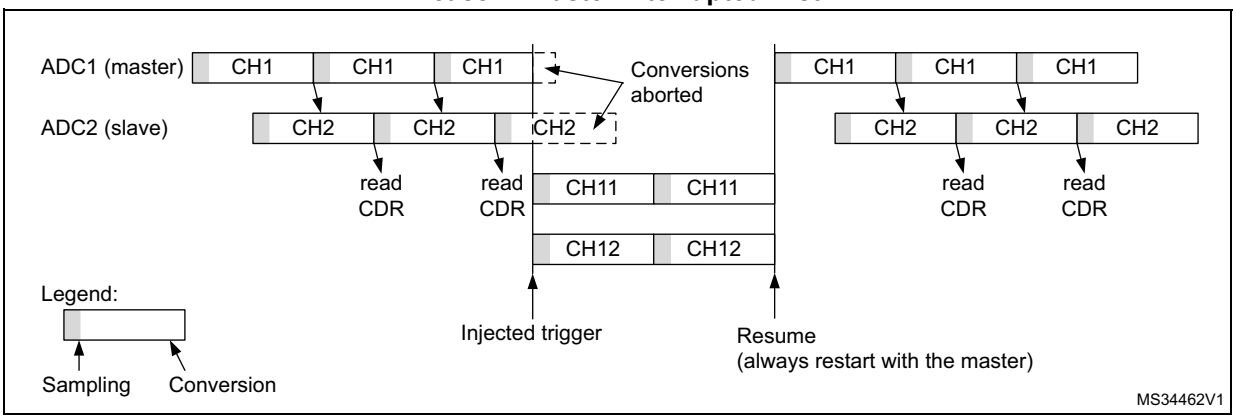
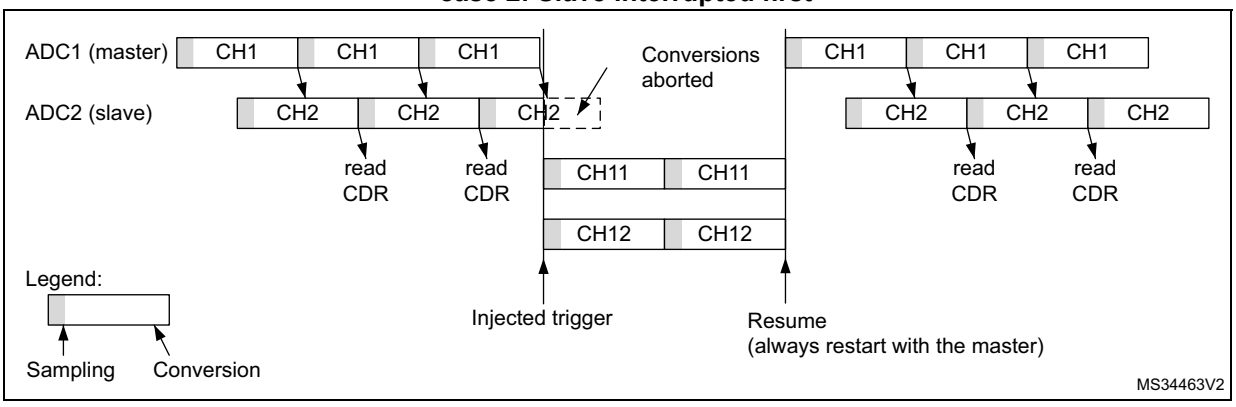


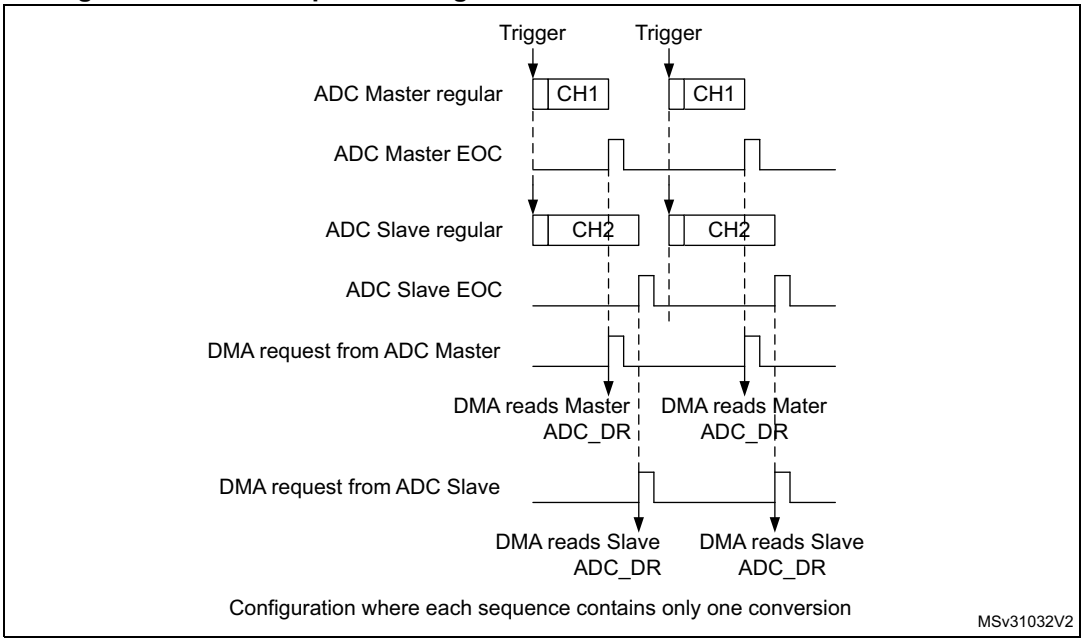
Figure 298. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 299: DMA Requests in regular simultaneous mode when MDMA = 0b00*).

Figure 299. DMA Requests in regular simultaneous mode when MDMA = 0b00



In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this MDMA bits must be configured in the ADCx_CCR register:

- **MDMA = 0b10**: A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCx_CDR contains the two half-words representing two ADC-converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.
  This mode is used in interleaved mode and in regular simultaneous mode when resolution is 10-bit or 12-bit.

  Example:

  Interleaved dual mode: a DMA request is generated each time 2 data items are available:

  1st DMA request: ADCx_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

  2nd DMA request: ADCx_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 300. DMA requests in regular simultaneous mode when MDMA = 0b10

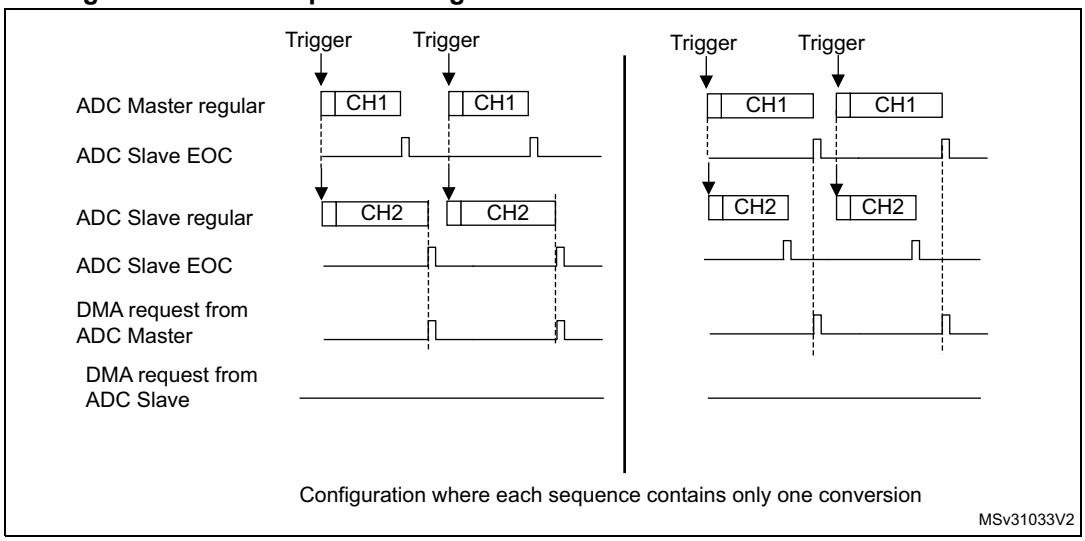
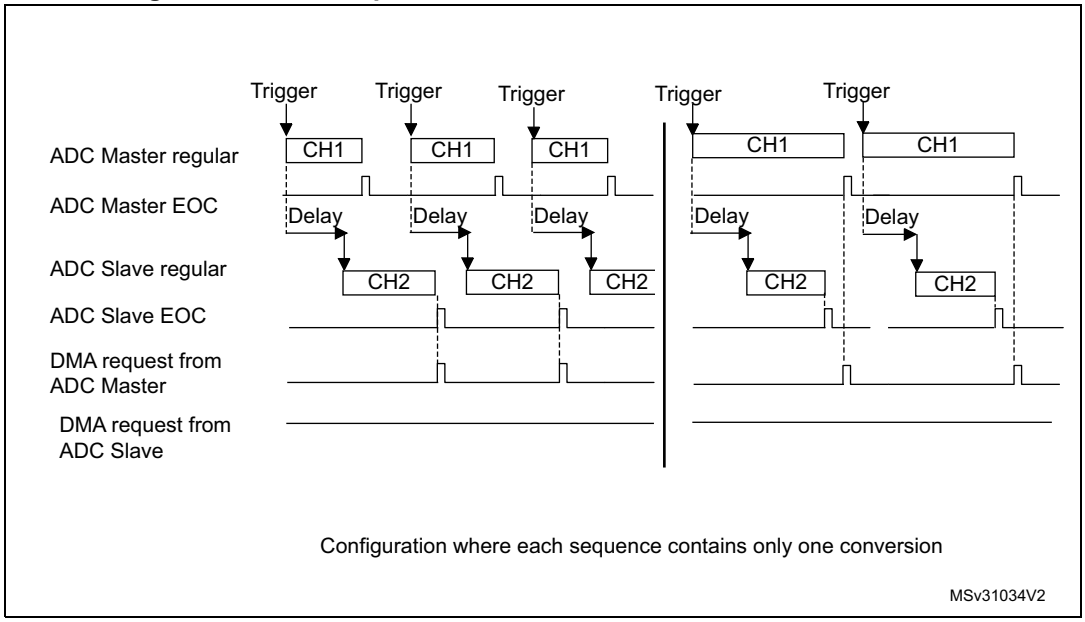


Figure 301. DMA requests in interleaved mode when MDMA = 0b10



Note: When using MDMA mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- **MDMA = 0b11**: This mode is similar to the MDMA = 0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.

  This mode is used in interleaved and regular simultaneous mode when resolution is 6-bit or when resolution is 8-bit and data is not signed (offsets must be disabled for all the involved channels).

  Example:

  Interleaved dual mode: a DMA request is generated each time 2 data items are available:

  1st DMA request: ADCx_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

  2nd DMA request: ADCx_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

Overrun detection

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the MDMA configuration). It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

DMA one shot mode/ DMA circular mode when MDMA mode is selected

When MDMA mode is selected (0b10 or 0b11), bit DMACFG of the ADCx_CCR register must also be configured to select between DMA one shot mode and circular mode, as explained in section *Managing conversions using the DMA* (bits DMACFG of master and slave ADC_CFGR are not relevant).

Stopping the conversions in dual ADC modes

The user must set the control bits ADSTP/JADSTP of the master ADC to stop the conversions of both ADC in dual ADC mode. The other ADSTP control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits ADSTART/JADSTART of the master and slave ADCs are both cleared by hardware.

29.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 336. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes
	End of conversion of a regular group	EOC	EOCIE		
	End of conversion sequence of a regular group	EOS	EOSIE		
	End of conversion of an injected group	JEOC	JEOCIE		
	End of conversion sequence of an injected group	JEOS	JEOSIE		
	Analog watchdog 1 status bit is set	AWD1	AWD1IE		
	Analog watchdog 2 status bit is set	AWD2	AWD2IE		
	Analog watchdog 3 status bit is set	AWD3	AWD3IE		
	End of sampling phase	EOSMP	EOSMPIE		
	Overrun	OVR	OVRIE		
	Injected context queue overflows	JQOVF	JQOVFIE		

29.6 ADC registers

Refer to *Section 1.2: Register conventions* for a list of abbreviations used in register descriptions.

Table 337. ADC global register map

Offset	Register
0x000 - 0x0B7	Master ADCx (ADC1 or ADC3)
0x0B8 - 0x0FF	Reserved
0x100 - 0x1B7	Slave ADCx (ADC2 or ADC4)
0x1B8 - 0x1FF	Reserved
0x200 - 0x2B7	Slave ADCx (ADC5)
0x2B8 - 0x2FF	Reserved
0x300 - 0x30F	Master and slave ADCs common registers (ADC1/2 or ADC3/4/5)

29.6.1 ADC register memory map (for each ADC)

Table 338. ADC register memory map

Offset	Register name
0x00	*ADC interrupt and status register (ADC_ISR)*
0x04	*ADC interrupt enable register (ADC_IER)*
0x08	*ADC control register (ADC_CR)*
0x0C	*ADC configuration register (ADC_CFGR)*
0x10	*ADC configuration register 2 (ADC_CFGR2)*
0x14	*ADC sample time register 1 (ADC_SMPR1)*
0x18	*ADC sample time register 2 (ADC_SMPR2)*
0x20	*ADC watchdog threshold register 1 (ADC_TR1)*
0x24	*ADC watchdog threshold register 2 (ADC_TR2)*
0x28	*ADC watchdog threshold register 3 (ADC_TR3)*
0x30	*ADC regular sequence register 1 (ADC_SQR1)*
0x34	*ADC regular sequence register 2 (ADC_SQR2)*
0x38	*ADC regular sequence register 3 (ADC_SQR3)*
0x3C	*ADC regular sequence register 4 (ADC_SQR4)*
0x40	*ADC regular data register (ADC_DR)*
0x4C	*ADC injected sequence register (ADC_JSQR)*
0x60 + 0x04 * (y -1), (y = 1 to 4)	*ADC offset y register (ADC_OFRy)*
0x80 + 0x04 * (y - 1), (y = 1 to 4)	*ADC injected channel y data register (ADC_JDRy)*
0xA0	*ADC Analog Watchdog 2 Configuration Register (ADC_AWD2CR)*

Table 338. ADC register memory map (continued)

Offset	Register name
0xA4	*ADC Analog Watchdog 3 Configuration Register (ADC_AWD3CR)*
0xB0	*ADC Differential mode Selection Register (ADC_DIFSEL)*
0xB4	*ADC Calibration Factors (ADC_CALFACT)*

29.6.1.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to *Section 29.4.20: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software. writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after ADC has been enabled (ADEN = 1) and when ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

29.6.1.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

29.6.1.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of ADC. Program first the bit ADCALDIF to determine if this calibration applies for Single-ended or Differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate ADC. Read at 1 means that a calibration in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the Single-ended or Differential inputs mode for the calibration.

0: Writing ADCAL launches a calibration in Single-ended inputs mode.

1: Writing ADCAL launches a calibration in Differential inputs mode.

Note: The software is allowed to write this bit only when ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Note: If the feature is not supported, this bit is reserved and kept at reset value.

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bits is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled

1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 29.4.5: ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and ADC injected sequence and triggers can be re-configured. ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set JADSTP only when JADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable ADC)

In Auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set ADSTP only when ADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable ADC).

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

In dual ADC regular simultaneous mode and interleaved mode, the bit ADSTP of the master ADC must be used to stop regular conversions. The other ADSTP bit is inactive.

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion immediately starts (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in Single conversion mode when software trigger is selected (JEXTSEL = 0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.

– in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable ADC).

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion immediately starts (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (EXTEN = 0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.

– in single conversion mode when HW trigger is selected (EXTEN != 0x0) and DMA is in one-shot mode (DMACFG = 0), once the DMA has reached the last DMA transfer, that is when the End of Transmission (DMA_EOT) occurs.

– in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable ADC)

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN = 1 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable ADC. ADC is effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable ADC.

Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0) except for bit ADVREGEN which must be 1 (and the software must wait for the startup time of the voltage regulator)

29.6.1.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALIGN	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL4	EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	RES[1:0]		Res.	DMACFG	DMAEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **JQDIS**: Injected Queue disable

These bits are set and cleared by software to disable the Injected Queue mechanism :

0: Injected Queue enabled

1: Injected Queue disabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).

A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1 (available on ADC1 and ADC3 only)

00001: ADC analog input channel 1 monitored by AWD1

.....

10010: ADC analog input channel 18 monitored by AWD1

others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: JSQR mode 0: The Queue is never empty and maintains the last written configuration into JSQR.

1: JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 29.4.20: Queue of context for injected conversions* for more information.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable Discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

It is not possible to use both auto-injected mode and Discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in Discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both Discontinuous mode and Continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

It is not possible to use both auto-injected mode and Discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to *Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)*.

0: Right alignment

1: Left alignment

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off

1: Auto-delayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both Discontinuous mode and Continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADCx_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)
01: Hardware trigger detection on the rising edge
10: Hardware trigger detection on the falling edge
11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0
00001: adc_ext_trg1
00010: adc_ext_trg2
00011: adc_ext_trg3
00100: adc_ext_trg4
00101: adc_ext_trg5
00110: adc_ext_trg6
00111: adc_ext_trg7
...
11111: adc_ext_trg31

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit
01: 10-bit
10: 8-bit
11: 6-bit

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMACFG**: Direct memory access configuration

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

0: DMA One Shot mode selected
1: DMA Circular mode selected

For more details, refer to *Managing conversions using the DMA*

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bit DMACFG of the ADCx_CCR register.

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA to manage automatically the converted data. For more details, refer to *Managing conversions using the DMA*.

0: DMA disabled

1: DMA enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bits MDMA[1:0] of the ADCx_CCR register.

29.6.1.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SMPTRIG	BULB	SWTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				OVSR[2:0]			JOVSE	ROVSE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled

1: Sampling time control trigger mode enabled

The sampling time starts on the trigger rising edge, and the conversion on the trigger falling edge.

EXTEN bit must be set to 01. BULB bit must not be set when the SMPTRIG bit is set.

When EXTEN bit is set to 00, set SWTRIG to start the sampling and clear SWTRIG bit to start the conversion.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 26 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

SAMPTRIG bit must not be set when the BULB bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Software trigger starts the conversion for sampling time control trigger mode

1: Software trigger starts the sampling for sampling time control trigger mode

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 24:17 Reserved, must be kept at reset value.

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular Oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 9 **TROVS**: Triggered Regular Oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bit field is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift

0001: Shift 1-bit

0010: Shift 2-bits

0011: Shift 3-bits

0100: Shift 4-bits

0101: Shift 5-bits

0110: Shift 6-bits

0111: Shift 7-bits

1000: Shift 8-bits

Other codes reserved

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:2 **OVSR[2:0]**: Oversampling ratio

This bit field is set and cleared by software to define the oversampling ratio.

000: 2x

001: 4x

010: 8x

011: 16x

100: 32x

101: 64x

110: 128x

111: 256x

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 1 **JOVSE**: Injected Oversampling Enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected Oversampling disabled

1: Injected Oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ROVSE**: Regular Oversampling Enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular Oversampling disabled

1: Regular Oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

29.6.1.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SMPPLUS	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SMPPLUS:** Addition of one clock cycle to the sampling time.

1: 2.5 ADC clock cycle sampling time becomes 3.5 ADC clock cycles for the ADC_SMPR1 and ADC_SMPR2 registers.

0: The sampling time remains set to 2.5 ADC clock cycles

To make sure no conversion is ongoing, the software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0.

Bit 30 Reserved, must be kept at reset value.

Bits 29:0 **SMP[9:0][2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 6.5 ADC clock cycles
010: 12.5 ADC clock cycles
011: 24.5 ADC clock cycles
100: 47.5 ADC clock cycles
101: 92.5 ADC clock cycles
110: 247.5 ADC clock cycles
111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

29.6.1.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:0 **SMP[18:10][2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 6.5 ADC clock cycles
010: 12.5 ADC clock cycles
011: 24.5 ADC clock cycles
100: 47.5 ADC clock cycles
101: 92.5 ADC clock cycles
110: 247.5 ADC clock cycles
111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

29.6.1.8 ADC watchdog threshold register 1 (ADC_TR1)

Address offset: 0x20

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AWDFILT[2:0]			LT1[11:0]											
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to *Section 29.4.27: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **AWDFILT**: Analog watchdog filtering parameter

This bit is set and cleared by software.

000: No filtering

001: two consecutive detection generates an AWDx flag or an interrupt

...

111: Eight consecutive detection generates an AWDx flag or an interrupt

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to *Section 29.4.27: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

29.6.1.9 ADC watchdog threshold register 2 (ADC_TR2)

Address offset: 0x24

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT2[7:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to *Section 29.4.27: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT2[7:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to *Section 29.4.27: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

29.6.1.10 ADC watchdog threshold register 3 (ADC_TR3)

Address offset: 0x28

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to *Section 29.4.27: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

29.6.1.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 1st in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion

0001: 2 conversions

...

1111: 16 conversions

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

29.6.1.12 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 9th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 8th in the regular conversion sequence

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 7th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 6th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 5th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

29.6.1.13 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 14th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 13th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 12th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 11th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 10th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

29.6.1.14 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 16th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 15th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

29.6.1.15 ADC regular data register (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RDATA[15:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 29.4.25: Data management*.

29.6.1.16 ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS = 0 (queue enabled), hardware and software trigger detection disabled. Otherwise, the queue is disabled as well as hardware trigger detection (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

If JQM = 1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 29.4.20: Queue of context for injected conversions)

Bits 6:2 **JEXTSEL[4:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

...

11111: adc_jext_trg31

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JL[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

29.6.1.17 ADC offset y register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y -1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSETy_EN	OFFSETy_CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OFFSETy[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OFFSETy_EN:** Offset y enable

This bit is written by software to enable or disable the offset programmed into bits OFFSETy[11:0].

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 30:26 **OFFSETy_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[11:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the data offset y.

If OFFSETy_EN is set, it is not allowed to select the same channel for different ADC_OFRy registers.

Bit 25 **SATEN**: Saturation enable

This bit is set and cleared by software to enable the saturation at 0x000 and 0xFFF for the offset function.

0: No saturation control, offset result can be signed

1: Saturation enabled, offset result unsigned and saturated at 0x000 and 0xFFF

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **OFFSETPOS**: Positive offset

This bit is set and cleared by software to enable the positive offset.

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSETy[11:0]:** Data offset y for the channel programmed into bits OFFSETy_CH[4:0]
These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.
Ex: if OFFSET1_CH[4:0] = 4 and OFFSET2_CH[4:0] = 4, this is OFFSET1[11:0] which is subtracted when converting channel 4.

29.6.1.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y - 1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **JDATA[15:0]:** Injected data
These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in *Section 29.4.25: Data management*.

29.6.1.19 ADC Analog Watchdog 2 Configuration Register (ADC_AWD2CR)

Address offset: 0xA0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[18:16]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **AWD2CH[18:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWD2CH[18:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the analog watchdog.

29.6.1.20 ADC Analog Watchdog 3 Configuration Register (ADC_AWD3CR)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[18:16]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **AWD3CH[18:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWD3CH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWD3CH[18:0] = 000..0, the analog Watchdog 3 is disabled

Note: The channels selected by AWD3CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the analog watchdog.

29.6.1.21 ADC Differential mode Selection Register (ADC_DIFSEL)

Address offset: 0xB0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[18:16]		
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIFSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **DIFSEL[18:0]**: Differential mode for channels 18 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as Single-ended or Differential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in Single-ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in Differential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (Single-ended input mode).

The software is allowed to write these bits only when ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

29.6.1.22 ADC Calibration Factors (ADC_CALFACT)

Address offset: 0xB4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **CALFACT_D[6:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new differential calibration is launched.

Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT_S[6:0]**: Calibration Factors In Single-ended mode

These bits are written by hardware or by software.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new single-ended calibration is launched.

Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

29.6.2 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

Table 339. ADC common registers

Offset	Register name
0x00	*ADCx common status register (ADCx_CSR) (x = 1/2 or 3/4/5)*
0x08	*ADCx common control register (ADCx_CCR) (x = 1/2 or 3/4/5)*
0x0C	*ADCx common regular data register for dual mode (ADCx_CDR) (x = 1/2 or 3/4/5)*

29.6.2.1 ADCx common status register (ADCx_CSR) (x = 1/2 or 3/4/5)

Address offset: 0x00

Reset value: 0x0000 0000

The above offset address is relative to the master ADC base address plus 0x300.

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **JQOVF_SLV:** Injected Context Queue Overflow flag of the slave ADC

This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC

This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17 **EOSMP_SLV:** End of Sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADC_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF_MST:** Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

29.6.2.2 ADCx common control register (ADCx_CCR) (x = 1/2 or 3/4/5)

Address offset: 0x08

Reset value: 0x0000 0000

The above address offset is relative to the master ADC base address plus 0x300.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESC[3:0]				CKMODE[1:0]	
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MDMA[1:0]		DMA CFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the clock to ADC. The clock is common for all ADCs.

0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
other: reserved

Note: The software is allowed to write these bits only when ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0). The ADC prescaler value is applied only when CKMODE[1:0] = 0b00.

Bits 17:16 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: adc_ker_ck (Asynchronous clock mode), generated at product level (refer to *Reset and clock control (RCC)*)
01: Reserved
10: adc_hclk/2 (Synchronous clock mode)
11: adc_hclk/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

Note: The software is allowed to write these bits only when ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 15:14 **MDMA[1:0]:** Direct memory access mode for dual ADC mode

This bit field is set and cleared by software. Refer to the DMA controller section for more details.

00: MDMA mode disabled
01: Reserved
10: MDMA mode enabled for 12 and 10-bit resolution
11: MDMA mode enabled for 8 and 6-bit resolution

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 13 **DMACFG**: DMA configuration (for dual ADC mode)

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

0: DMA One Shot mode selected
1: DMA Circular mode selected

For more details, refer to *Managing conversions using the DMA*

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY:** Delay between 2 sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 340* for the value of ADC resolution versus DELAY bits values.

Note: The software is allowed to write these bits only when ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode.

All ADCs independent:
00000: Independent mode

00001 to 01001: Dual mode, master and slave ADCs working together
00001: Combined regular simultaneous + injected simultaneous mode
00010: Combined regular simultaneous + alternate trigger mode
00011: Combined Interleaved mode + injected simultaneous mode
00100: Reserved
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only

All other combinations are reserved and must not be programmed

Note: The software is allowed to write these bits only when ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Table 340. DELAY bits versus ADC resolution

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$
0001	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$
0010	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$

Table 340. DELAY bits versus ADC resolution (continued)

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0011	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$
0100	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$
0101	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0110	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0111	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1000	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1001	10 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1010	11 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1011	12 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
others	12 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$

29.6.2.3 ADCx common regular data register for dual mode (ADCx_CDR) (x = 1/2 or 3/4/5)

Address offset: 0x0C (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_SLV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA_MST[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 29.4.29: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 29.4.29: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)*)

In MDMA = 0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

30 Sigma-delta analog-to-digital converter (SDADC) digital interface

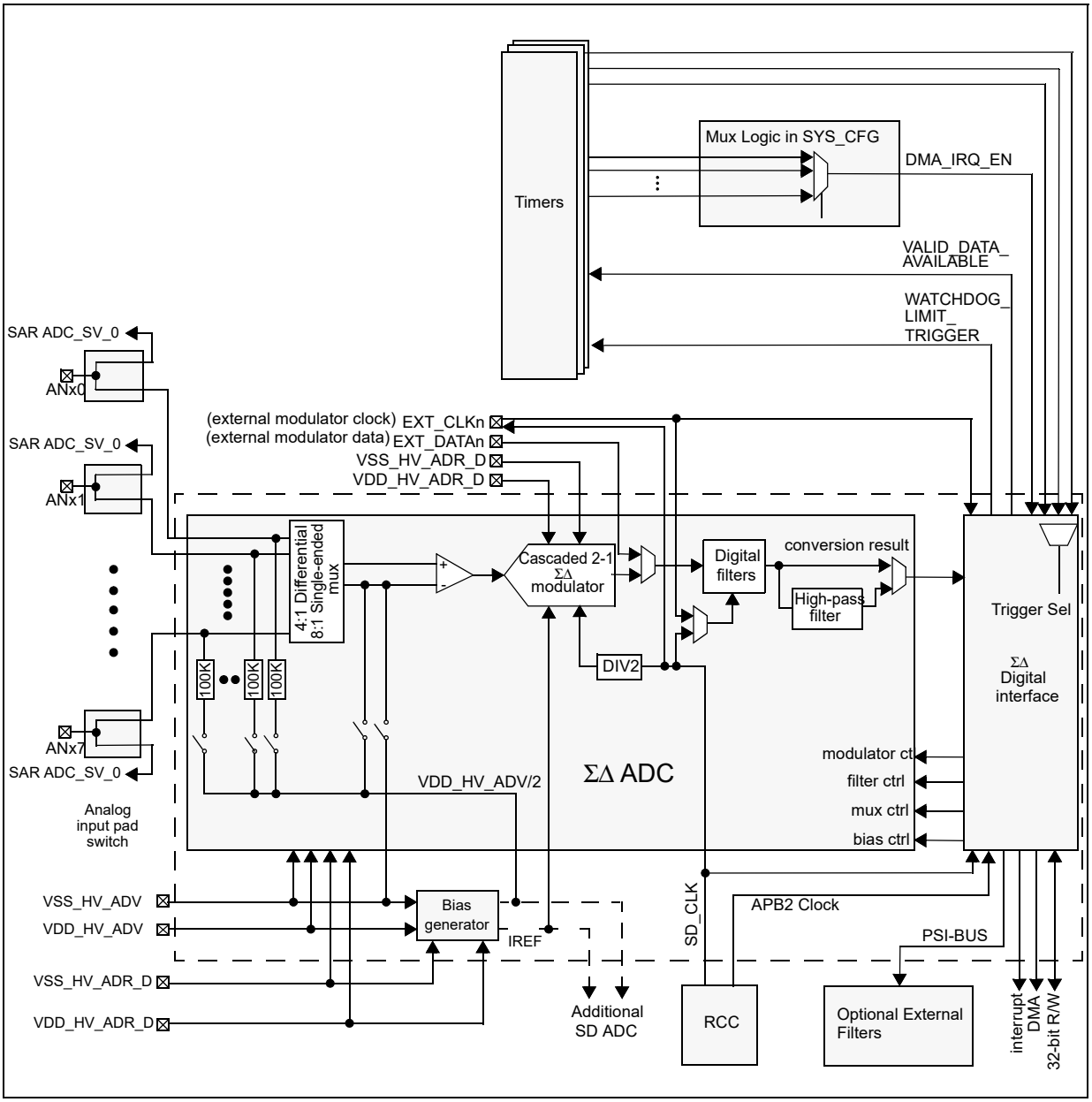
30.1 Introduction

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides accurate conversion data for a wide range of applications.

30.2 Overview

Figure 302 depicts the block diagram of an SDADC and its digital control block. The SDADC digital interface block provides all control information for operating mode selection (single-ended or differential), analog input gain, decimation rate, high-pass filter enable, and so on. It also generates the bias controls for common mode voltage selection, to bias the positive and negative terminals to different voltage levels (for example, half scale bias, reference ground, reference supply) for diagnostics. It generates the analog multiplexer control signals for the selected channels. It also provides external modulator support and keeps the internal modulator in low consumption mode. To assist synchronous operation across multiple ADC blocks, hardware trigger support is implemented to start data conversion.

Figure 302. SDADC block diagram



30.3 Features

- 16-bit data resolution output
- Single-ended input or differential input mode of operation
- Programmable wraparound mechanism for both modes of operation
  - Configurable trigger sources: hardware or software
  - Configurable initial entry and wraparound values for the loop
- Configurable biasing for negative input terminal in single-ended mode
- Gain and offset calibration support using fixed bias for input terminals
- Programmable decimation rate
- Cascaded COMB + FIR decimation filter with an optional FIR filter for the SD ADC data path
- Programmable gain for analog inputs
- Optional external modulator support
- Optional high-pass filter for pure AC application
- Hardware trigger support for synchronous operation of multiple SDADC blocks
- Trigger event output generation by software
- Programmable FIFO structure for storing 16-bit converted data
- Left/right data alignment
- Signed/unsigned format of converted data output
- Interrupt/DMA request generation based on various conditions:
  - Data buffer at or above threshold
  - Data buffer overrun
  - WDG crossover event
- Programmable threshold range WDG support
- Low consumption mode
- Optional conversion process halt mechanism on SoC debug request
- Externally supplied timestamp added to data samples

30.4 Modes of operation

This section describes the different operation modes of the SDADC. When exiting reset, the mode of operation is determined by the Module Configuration Register (MCR).

30.4.1 Differential input mode

This mode is entered by negating the MCR[MODE] bit. In this mode, a pair of analog inputs are connected to positive and negative terminals of ADC. In order to support gain and offset calibration for diagnostics, it is possible in this mode to select fixed bias voltages to both input terminals.

30.4.2 Single-ended input mode

This mode is entered by asserting the MCR[MODE] bit. In this mode, the negative input terminal is biased with a fixed voltage based on selection information in MCR, and the other analog input is connected to the selected external analog channel.

30.4.3 External modulator mode

This mode is entered by asserting the MCR[EMSEL] bit. In this mode, the data stream and clock outputs from the external modulator available on BSEXT/CLKEXT input pins are routed directly to the SDADC block. The internal modulator is bypassed in this mode. Only the digital filters of the SDADC are used for result calculation. The source for the external modulator clock can be the internal on-chip clock source or the direct clock output coming from the external modulator. The external pin function control for these pins must be configured in the integration logic on a device, and is not included in the SDADC digital interface.

30.5 External signal description

Table 341 shows the list of signals driven by external pins. At the chip integration level, some of the digital and analog signals might share pins or may not be available external to the chip.

Table 341. Signal properties

Name	Function	I/O	Reset	Pull-up
AIN[0:7]	Single-ended analog inputs or Differential pairs of analog inputs	I	—	—
BSEXT	External modulator data input	I	0	—
CLKEXT	External modulator clock input	I	0	—
VREFP = VDD_HV_ADR_D	Voltage reference for SDADC	I	—	—
VREFN = VSS_HV_ADR_D	Ground reference for SDADC	I	—	—
VDD_HV_ADV	Analog voltage supply for SDADC	I	—	—
VSS_HV_ADV	Analog ground supply for SDADC	I	—	—
VDD_LV	Digital voltage supply for SDADC	I	—	—
VSS_LV	Digital ground supply for SDADC	I	—	—

30.5.1 Detailed signal descriptions

Table 342 provides a detailed description of the external signals.

Table 342. Detailed signal descriptions

Signal	I/O	Description	
AIN[0:7]	I	Each analog input can be used as a single-ended analog input or one of a differential pair of analog inputs which is connected to either positive or negative terminals of the SDADC, depending on the mode of operation and polarity selection.	
		State meaning	Analog input
		Timing	Can be sampled at any time by the ADC.
BSEXT	I	Data stream input provided by external modulator.	
		State meaning	Asserted—Data input is 1 Negated—Data input is 0
		Timing	Assertion—May occur at any time, but data stream is valid only when external modulator is selected by MCR[MODSEL] = 1. Negation—Ignored when external modulator is not selected.
CLKEXT	I	Clock input provided by external modulator.	
		State meaning	Asserted—clock high level Negated—clock low level
		Timing	Assertion—May occur at any time, but clock is valid only when external modulator is selected by MCR[MODSEL] = 1. Negation—Ignored when external modulator is not selected.

30.6 Memory map and register descriptions

30.6.1 Memory map

This section provides the memory map for the SDADC block.

Table 343. SDADC memory map

Address Offset	Register name	Width (in bits)	Access	Reset value	Section
0x0000	Module Configuration Register (SDADC_MCR)	32	R/W	0x0000_0000	*Section 30.6.2.1*
0x0004	Channel Selection Register (SDADC_CSR)	32	R/W	0x0000_0000	*Section 30.6.2.2*
0x0008	Reset Key Register (SDADC_RKR)	32	R/W	0x0000_A50F	*Section 30.6.2.3*
0x000C	Status Flag Register (SDADC_SFR)	32	R/W	0x0000_0100	*Section 30.6.2.4*
0x0010	Request Select and Enable Register (SDADC_RSER)	32	R/W	0x0000_0000	*Section 30.6.2.5*
0x0014	Output Settling Delay Register (SDADC_OSDR)	32	R/W	0x0000_0000	*Section 30.6.2.6*
0x0018	FIFO Control Register (SDADC_FCR)	32	R/W	0x0000_0006	*Section 30.6.2.7*
0x001C	Software Trigger Key Register (SDADC_STKR)	32	R/W	0x0000_0000	*Section 30.6.2.8*
0x0020	Converted Data Register (SDADC_CDR)	32	R	0x0000_0000	*Section 30.6.2.9*
0x0024	WDG Threshold Register (SDADC_WTHHLR)	32	R/W	0x0000_0000	*Section 30.6.2.10*

Table 343. SDADC memory map (continued)

Address Offset	Register name	Width (in bits)	Access	Reset value	Section
0x0028	Latest Data Register (SDADC_LDREG)	32	R	0x0000_0000	*Section 30.6.2.11*
0x002C	Latest Time-Stamp Register (SDADC_LTREG)	32	R	0x0000_0000	*Section 30.6.2.12*

30.6.2 Registers description

30.6.2.1 Module Configuration Register (SDADC_MCR)

The Module Configuration register (MCR) consists of different control bits to select the operating modes and various configuration settings which define the behavior of the SDADC block.

Offset 0x0000 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EXTFILTER	LTMW	TRIG_SRC_ADC	PDR					0	PGAN			ODF	ODA	EMSEL	HPFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WDGEN	TRIGEDSEL		TRIGEN	TRIGSEL				FRZ	BANDSEL	FSEL	VCOMSEL	WRMODE	GECEN	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 303. Module Configuration Register (SDADC_MCR)

Table 344. SDADC_MCR field descriptions

Field	Description
31 EXTFILTER	External Filter selection 0 Only Internal Filter (Finished and valid data) 1 External Filter (Raw data after COMB filter to be processed by external filter)
30 LTMW	Level Trigger Mode selection for Watchdog 0 Level trigger mode 1 Hysteresis mode
29 TRIG_SRC_ADC	HW trigger source selection 1 Trigger source is SD-ADC 0 Trigger source is not SD-ADC

Table 344. SDADC_MCR field descriptions (continued)

Field	Description
28:24 PDR	Programmable Decimation Rate This field selects the over-sampling ratio to be applied to support different pass-bands with a fixed input sampling clock. The output data rate f_d is equal to f_s / (2×OSR), where f_s is the input sampling clock frequency. When external modulator is selected, the output data rate is independent of this field and is fixed to f_s / 256. 00000 OSR = 24 00001 OSR = 28 00010 OSR = 32 00011 OSR = 36 00100 OSR = 40 00101 OSR = 44 00110 OSR = 48 00111 OSR = 56 01000 OSR = 64 01001 OSR = 72 01010 OSR = 75 01011 OSR = 80 01100 OSR = 88 01101 OSR = 96 01110 OSR = 112 01111 OSR = 128 10000 OSR = 144 10001 OSR = 160 10010 OSR = 176 10011 OSR = 192 10100 OSR = 224 10101 OSR = 256 10110 OSR = 512 10111 OSR = 1024 11000-11111 Reserved
22:20 PGAN	Programmable Gain This field selects the gain to be applied to the analog input stage of the SDADC. The effective analog input becomes the input voltage level multiplied by the gain factor. **Note:** When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0. 000 Gain = 1 001 Gain = 2 010 Gain = 4 011 Gain = 8 100 Reserved 101 Reserved 110 Reserved 111 Gain = 16
19 ODF	Output Data Format ODF selects the representation for digital part. 0 Output data is unsigned 1 Output data is signed and sign extended to 16 bits

Table 344. SDADC_MCR field descriptions (continued)

Field	Description
18 ODA	Output Data Alignment 0 Output data is right-aligned 1 Output data is left-aligned
17 EMSEL	External Modulator Selection 0 External modulator data and clock inputs are ignored 1 External modulator data stream and clock inputs are provided to SDADC
16 HPFEN	High Pass Filter Enable 0 High-pass (DC removal) filter is disabled 1 High-pass (DC removal) filter is enabled
15 WDGEN	Watchdog Enable This bit enables the WDG monitor. 0 WDG disabled 1 WDG is enabled
14:13 TRIGEDSEL	Trigger Edge Selection 00 Falling edge of trigger input is selected 01 Rising edge of trigger input is selected 10 Both edges of trigger input are selected 11 Both edges of trigger input are selected
12 TRIGEN	Trigger Enable This field enables the hardware trigger input to initiate a fresh conversion. Upon receiving a trigger event, SDADC reset input is asserted and deasserted synchronously with respect to the peripheral clock. The reset pulse width is fixed to four peripheral clock cycles. The trigger event is ignored if the SDADC internal modulator is not enabled (MCR[EN] = 0) or the external modulator is selected (MCR[EMSEL] = 1). 0 Trigger input is disabled 1 Trigger input is enabled
11:8 TRIGSEL	Trigger Input Selection This field selects which input is used for hardware-triggered conversions. 0000 SD_ADC_1 sw_trig_out 0001 SD_ADC_2 sw_trig_out 0010 Timer tim1_trgo 0011 Timer tim1_trgo2 0100 Timer tim8_trgo 0101 Timer tim8_trgo2 0110 Timer tim2_trgo 0111 Timer tim3_trgo 1000 Timer tim4_trgo 1001 Timer tim15_trgo 1010 Timer hrtim1_adctrig1 1011 Timer hrtim1_adctrig2 1100 Timer hrtim1_adctrig3 1101 Timer hrtim2_adctrig1 1110 Timer hrtim2_adctrig2 1111 Timer hrtim2_adctrig3

Table 344. SDADC_MCR field descriptions (continued)

Field	Description
7 FRZ	Freeze This field enables stopping the SDADC conversions at the end of the current channel conversion when the SoC enters debug mode. When the SoC enters debug mode, further conversions by the SDADC analog block are stopped, and converted data output from the SDADC analog block are ignored. The ongoing channel conversion is completed and the converted data is stored. When the SoC exits debug mode, SDADC resumes the job that was left before halting the conversions. 0 Conversions are not stopped 1 Conversions are stopped
6 BANDSEL	Output Data Rate Bandwidth Selection 0 1/3 (default) 1 1/6
5 FSEL	ADC Internal Filter Selection 0 Comb + FIR (default) 1 Comb alone
4 VCOMSEL	Common Voltage Bias Selection VCOMSEL selects the reference for analog part. This field selects the common voltage bias for the negative input terminal of the SDADC during single-ended mode (MODE = 1). 0 Negative input terminal is biased with VREFN 1 Negative input terminal is biased with VREFP/2 (half-scale bias)
3 WRMODE	Wrap-Around Mode This bit selects the wraparound mechanism for conversion of programmed sequence of channels. 0 Wraparound mechanism disabled (in this case the default software control mechanism is enabled) 1 Wraparound mechanism enabled
2 GECEN	Accurate Gain Error Mode Enable This bit selects the accurate gain error mode, and must be set during gain error calibration as well as during conversions requiring accurate gain error mode. It can be kept '0' if gain error calibration mode is not needed. 0 Gain error calibration mode disabled 1 Gain error calibration mode enabled
1 MODE	Mode selection 0 Differential input mode selected 1 Single-ended input mode selected
0 EN	Enable for SDADC block Refer to *Section 30.8: Initialization information* for details on configuration sequence. 0 SDADC internal modulator placed in low consumption mode 1 SDADC internal modulator enabled

30.6.2.2 Channel Selection Register (SDADC_CSR)

The Channel Selection Register (CSR) selects analog inputs AIN[0:7] to be connected to positive and negative terminals of the SDADC and controls the biasing of analog inputs to half-scale bias through 100 Kohm resistance.

Offset 0x0004 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	BIASEN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	ANCHSEL_WRAP			0	0	0	0	0	ANCHSEL		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 304. Channel Selection Register (SDADC_CSR)

Table 345. SDADC_CSR field descriptions

Field	Description
23:16 BIASEN	Bias enable If BIASEN[i] = 1, analog input AIN[i] (could be single-ended input or differential analog input) is connected to half-scale bias through a 100 KΩ resistor.
10:8 ANCHSEL_WRAP	Analog Channel Selection Wraparound Value (ANCHSEL_WRAP) When in wraparound mode, this field indicates the maximum value of the wraparound counter, after which it wraps around back to initial value 000. This value must be programmed before enabling wraparound mechanism (MCR[WRMODE] = 1).
2:0 ANCHSEL	Analog Channel Selection (ANCHSEL) Based on single-ended or differential mode of operation (MCR[MODE]), common mode voltage selection (MCR[VCOMSEL]), this field defines the connectivity of analog inputs to either positive or negative polarity terminals of the SDADC according to *Table 346*. Wraparound Mode: – This field also indicates the initial entry value for the first loop of the wraparound sequence from where the wraparound counter starts incrementing when wraparound mode is entered (when wraparound mechanism enabled (MCR[WRMODE] = 1)). – If this field is written any time during wraparound mode (when MCR[WRMODE] = 1), the present value of the wraparound counter is updated with the value of ANCHSEL — the next trigger acts on the updated channel, and the next increment is on the updated channel. Here, the value of ANCHSEL chosen must be < ANCHSEL_WRAP; otherwise it is ignored by hardware. – Wraparound is always from initial counter value 000.

Table 346. Analog input AIN[0:7] selection

MODE	VCOMSEL	ANCHSEL	INP (Positive terminal)	INM (Negative terminal)
1	0	000	AIN[0]	VREFN
		001	AIN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AIN[0]	VREFP/2
		001	AIN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AIN[0]	AIN[1]
		001	Reserved	Reserved
		010	Reserved	Reserved
		011	Reserved	Reserved
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

30.6.2.3 Reset Key Register (SDADC_RKR)

The Reset Key Register (RKR) is the target for a predefined key write operation used to reset the SDADC to start a fresh conversion.

Offset 0x0008 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESET_KEY															
W																
Reset	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

Figure 305. Reset Key Register (SDADC_RKR)

Table 347. SDADC_RKR field descriptions

Field	Description
15:0 RESET_KEY	Reset Key This field, when written with 0x5AF0, is used to generate the reset for the SDADC block and to reload the internal counter with the start value programmed in OSDR. Any write access to this register which is different from the predefined key is ignored. Read access always returns the inverted key.

30.6.2.4 Status Flag Register (SDADC_SFR)

The Status Flag Register (SFR) contains status and flag bits. These bits are set by hardware and reflect the status of the SDADC and indicate the occurrence of events that may generate an interrupt or a DMA request. Software can clear some flags by writing ‘1’ to those bits. Writing a ‘0’ to a flag has no effect.

Offset 0x000C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	ANCHSEL_CNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	DFEF	0	0	0	WTHH	WTHL	CDVF	DFORF	DFFF
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 306. Status Flag Register (SDADC_SFR)

Table 348. SDADC_SFR field descriptions

Field	Description
18:16 ANCHSEL_CNT	Analog Channel Selection Counter (ANCHSEL_CNT) This field reflects the current value of the internal Analog Channel Selection Counter. It indicates the present number of the channel selected through the analog multiplexer.
8 DFEF	Data FIFO Empty Flag This field is asserted when there is no data word in the FIFO. 0 Data FIFO is not empty. 1 Data FIFO is empty.
4 WTHH	Watchdog Upper Threshold Cross Over Event This field is set when Converted Data crosses Upper Threshold of WDG. This bit automatically gets cleared on arrival of ipd_done when current ongoing DMA transfer completes, It can also be cleared by software writing ‘1’ to clear. 0 Watchdog Upper Threshold Cross Over Event did not occur. 1 Watchdog Upper Threshold Cross Over Event occurred.
3 WTHL	Watchdog Lower Threshold Cross Over Event This field is set when Converted Data crosses Lower Threshold of WDG. This bit automatically gets cleared on arrival of ipd_done when current ongoing DMA transfer completes. It can be cleared by software writing ‘1’ to clear. 0 Watchdog Lower Threshold Cross Over Event did not occur. 1 Watchdog Lower Threshold Cross Over Event occurred.
2 CDVF	Converted Data Valid Flag This field gives the actual status of data coming from the SDADC block. The converted data is not valid or becomes corrupted whenever there is a change in configuration settings determined by MCR, channel selection programmed in CSR. A write access to the registers MCR and CSR resets this flag. This flag is set automatically after generating SDADC reset and internal timer counts OSD cycles of output clock f_d. 0 Data output from SDADC is not valid. 1 Data output from SDADC is valid.

Table 348. SDADC_SFR field descriptions (continued)

Field	Description
1 DFORF	Data FIFO Overrun Flag This field indicates that software or DMA failed to prevent FIFO from overflowing with converted data words. Software can clear this bit by writing '1' to it after reading the FIFO contents through CDR. This flag remains set if software or DMA failed to clear the condition that caused this flag setting. If DFORF is set, further data words are not received into FIFO even if sufficient room exists. 0 No overrun has occurred since the last time the flag was cleared. 1 Overrun has occurred or the DFORF has not been cleared since the last overrun occurred.
0 DFFF	Data FIFO Full Flag This bit is set when the number of converted data words in the FIFO is equal to or greater than the number of data words indicated by FCR[FTHLD] value. Software can clear this bit by writing '1' to it. This bit is also cleared by an acknowledgment from the DMA controller when DMA request generation is selected. This bit remains set if software or DMA failed to clear the condition that caused this flag setting. Even if DFFF is set, data words continue to be received until an overrun condition occurs. Data words are not received if the converted data is not valid (CDVF = 0). 0 The number of data words in FIFO is less than or equal to the number indicated by FCR[FTHLD]. 1 The number of data words in FIFO is greater than the number indicated by FCR[FTHLD] at some point in time since this flag was last cleared. **Note:** Whenever MCR[EN] is cleared in order to stop the SDADC or change the channel configuration, it is necessary to clear the RSER[DFFDIRE] bit as well. Similarly if DMA or software is too slow in not reading the FIFO data words resulting in DFFF condition it is advisable to stop the SDADC by clearing the MCR[EN] followed by clearing of RSER[DFFDIRE] bit. Both of these actions ensure safe operation.

30.6.2.5 Request Select and Enable Register (SDADC_RSER)

The Request Select and Enable Register (RSER) serves two purposes:

- Enables the flag bits in SFR to generate DMA requests or interrupt requests.
- Selects the type of request to generate.

Note: This register may be read at any time, but must only be written when SDADC is disabled (MCR[EN] = 0).

Refer to the field descriptions for the type of requests that are supported.

Offset 0x0010 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WTHDIRS	DFFDIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GGE	DIRFWGS	SGARE	CLERE	CTRE	0	0	0	0	0	0	0	WTHDIRE	CDVEE	DFORIE	DFFDIRE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 307. Request Select and Enable Register (SDADC_RSER)

Table 349. SDADC_RSER field descriptions

Field	Description
17 WTHDIRS	WDG Threshold Cross Over Event DMA/Interrupt Request Select When SFR[WTHH/WTHL] and RSER[WTHDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected. 1 DMA request is selected.
16 DFFDIRS	Data FIFO Full DMA/Interrupt Request Select When the SFR[DFFF] and RSER[DFFDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected. 1 DMA request is selected.
15 GGE	Global Gating Enable Whenever Global Gating is enabled, generation of DMA/Interrupt request, write to FIFO and data stream through PSI Interface is dependent upon the external gating signal. 0 Gating feature disabled. 1 Gating feature enabled.
14 DIRFWGS	DMA/Interrupt or FIFO Write Gating Select Whenever Global DMA/Interrupt gating feature is enabled, this field determines the gating functionality as follows: 0 DMA/Interrupt request is gated. 1 Write to FIFO is gated.
13 SGARE	Synchronized Gate Assert Request Enable 0 Global gate signal assertion is not synchronized. 1 Global gate signal assertion is synchronized with the new data sample.

Table 349. SDADC_RSER field descriptions (continued)

Field	Description
12 CLERE	Capture Last Event Request Enable Whenever Global Gating feature is enabled (GGE - 1) and write to FIFO is gated (DIRFWGS - 1), this field determines whether to copy the last data word/timestamp to FIFO or not, once gating window is opened. 0 Don't copy the last data word/timestamp. 1 Copy the last data word and timestamp if capturing timestamp is enabled.
11 CTRE	Capture TimeStamp Request Enable This field selects whether to capture timestamp or not. 0 Don't capture the timestamp. 1 Capture the timestamp information inside FIFO.
3 WTHDIRE	WDG Threshold Cross Over Event DMA/Interrupt Request Enable This field enables the SFR[WTHH/WTHL] field to generate a request. The WTHDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt/DMA request is disabled on WDG Threshold Cross Over Event. 1 Interrupt/DMA request is enabled on WDG Threshold Cross Over Event.
2 CDVEE	Conversion Data Valid Event Enable The status of SFR[CDVF] gated with RSER[CDVEE] is passed on as an event on “conversion_flag” output port, which can be used at SoC level for indication of valid data conversion window. Once enabled, the event output remains asserted as long as valid data conversions are ongoing. 0 Event output disabled. 1 Event output assertion/deassertion based on the SFR[CDVF] field.
1 DFORIE	Data FIFO Overrun Interrupt Enable This bit enables the SFR[DFORF] field to generate an interrupt request. The final interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt request is disabled when data FIFO overrun condition occurs. 1 Interrupt request is enabled when data FIFO overrun condition occurs.
0 DFFDIRE	Data FIFO Full DMA/Interrupt Request Enable This field enables the SFR[DFFF] field to generate a request. The DFFDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field. To have safe operation this bit must not be set when MCR[EN]=0. Whenever MCR[EN] is cleared this bit must also be cleared if already set and FCR[FRST] can be set to clear the previous FIFO content. 0 Interrupt/DMA request is disabled when data FIFO full condition occurs. 1 Interrupt/DMA request is enabled when data FIFO full condition occurs.

30.6.2.6 Output Settling Delay Register (SDADC_OSDR)

The Output Settling Delay Register (OSDR) provides a delay value to qualify the converted output data coming from the SDADC.

Offset 0x0014 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	OSD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 308. Output Settling Delay Register (SDADC_OSDR)

Table 350. SDADC_OSDR field descriptions

Field	Description
7:0 OSD	Output Settling Delay This field defines the delay to qualify the conversion data stored in CDR. Whenever the SDADC block is reset in order to start the conversion from a fresh state, an internal timer is loaded with this start value. The counter counts down with output clock f_d until it reaches zero, and then it generates a flag which qualifies the converted data. The output clock f_d is equal to f_s / (2 x OSR), where f_s is the input sampling clock frequency. When external modulator is selected, the output clock is fixed to f_s / 256. See the electrical specifications for the settling delays.

30.6.2.7 FIFO Control Register (SDADC_FCR)

The FIFO Control Register (FCR) provides the ability to enable or disable FIFO functionality, and to control the amount of available FIFO depth to be used.

Note: It is important to note that the FIFO depth is set at the factory and cannot be changed but the amount of available FIFO space actually used can be set by specifying the value in the FTHLD field. The FSIZE field indicates the amount of FIFO space available. By setting the value of FTHLD to be less than the FIFO depth indicated by FSIZE, the FIFO is considered to be full when the number of data words in the FIFO is equal to or greater than FTHLD.

This register may be read at any time, but must only be written when SDADC is disabled (MCR[EN] = 0).

Offset 0x0018 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																FRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	FTHLD				0	0	0	0	FOWEN	FSIZE		FE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 309. FIFO Control Register (SDADC_FCR)

Table 351. SDADC_FCR field descriptions

Field	Description
16 FRST	FIFO Flush Reset This is functional soft reset to flush the current FIFO content and initialize the internal FIFO pointers. This is W1S (write one shot) bit which can be written as ‘1’ but always read as ‘0’. Use this option only after disabling the ADC that is MCR[EN]=’0’. 0 No effect. 1 Generate a single cycle reset event to flush the FIFO.
11:8 FTHLD	FIFO Threshold It sets the threshold to generate DMA/Interrupt event. For example: FTHLD = "0000" indicates 1 data word, FTHLD = "0011" indicates 4 data words and so on. When Time Stamp feature is enabled by asserting RSER[CTRE] bit, then FTHLD = "0000" indicates 1 data word along with their time stamp, FTHLD = "0011" indicates 4 data words along with their time stamp and so on. When Time Stamp feature is disabled. then FTHLD = "0000" indicates 1 data word, FTHLD = "0011" indicates 4 data words and so on. When the number of data words in the data FIFO is greater than or equal to the number of data words indicated by the FTHLD field, the FIFO full event is flagged. An interrupt or a DMA request is generated as determined by DFFDIRE and DFFDIRS fields of RSER. For proper operation, number of data words indicated by the FTHLD field must be set less than or equal to the FIFO size as indicated by FSIZE.
3 FOWEN	FIFO Over Write Enable When set the built-in FIFO structure allows the overwriting of new converted data over older converted data after the FIFO has number of entries equal to the FSIZE. In this case overrun is not seen. 0 Data FIFO OW option disabled. 1 Data FIFO OW option enabled.

Table 351. SDADC_FCR field descriptions (continued)

Field	Description
2:1 FSIZE	FIFO Size The maximum number of converted datawords that can be stored in the data FIFO before an overrun occurs. This field is read-only. The reset value of this field depends on the implementation parameter to define the FIFO size. 00 FIFO depth = 1 data word 01 FIFO depth = 4 data words 10 FIFO depth = 8 data words 11 FIFO depth = 16 data words
0 FE	FIFO Enable When this field is set the built-in FIFO structure is enabled. The size of the FIFO structure is indicated by the FSIZE field. If this field is not set then the FIFO depth is set to one data word regardless of the value in the FSIZE field. 0 Data FIFO is not enabled for multiple data words; FIFO depth is one data word. 1 Data FIFO is enabled; FIFO depth is indicated by FSIZE.

30.6.2.8 Software Trigger Key Register (SDADC_STKR)

The Software Trigger Key Register (STKR) allows generating the trigger event output by a software write operation.

Offset 0x001C — Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ST_KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 310. Software Trigger Key Register (SDADC_STKR)

Table 352. SDADC_STKR field descriptions

Field	Description
15:0 ST_KEY	Software Trigger Key This bit field, when written with 0xFFFF, is used to generate a trigger event output which can be used to trigger conversions of multiple SDADC blocks synchronously depending on the SoC implementation. Any write access to this register which is different from the predefined key is ignored. Read access always returns 0x0000.

30.6.2.9 Converted Data Register (SDADC_CDR)

The Converted Data Register (CDR) stores the converted data in the data FIFO.

Note: When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0.

Offset 0x0020 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	CDATA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 311. Converted Data Register (SDADC_CDR)

When timestamp is read, all 24 bits are valid. When converted data is read, 16 bits are valid.

If the output data format is right-aligned (ODA is reset), then the register bit map is as follows.

Table 353. SDADC_CDR field descriptions

Field	Description
23:16 CDATA	All bits are read as 0
15:0 CDATA	Converted Data Register The converted data words can be read from FIFO by reading this register. The data width is 16 bits which has actual 16-bit data (bits 15 down to 0) coming from ADC. If the data width is less than 16 (for example: 14) and if ODF is set, then: – If bit 13 is ‘1’, then the result is negative and bits 15 and 14 also return ‘1’. – If bit 13 is ‘0’, then the result is positive and bits 15 and 14 return ‘0’. If ODF is reset, then bits 15 and 14 always return ‘0’.

Note: If the output data format is left-aligned (ODA is set), then the register bit map is as follows.

Table 354. SDADC_CDR field descriptions

Field	Description
23:16 CDATA	All bits are read as 0
15:0 CDATA	Converted Data Register The converted data words can be read from FIFO by reading this register. If the actual data width is less than 16 bits (for example: 14), then bits 15 down to 2 store the data coming from SDADC and bits 1 and 0 always return ‘0’.

30.6.2.10 WDG Threshold Register (SDADC_WTHHLR)

This is the Converted Data Range WDG Threshold register.

Offset: 0x0024 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	THRH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	THRL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 312. WDG Threshold Register (SDADC_WTHHLR)

Note: A single WDG monitor checks for lower and upper threshold crossover events.

Table 355. SDADC_WTHHLR field descriptions

Field	Description
31:16 THRH	WDG Upper Threshold Value This indicates the upper threshold for the converted data on any channel, above which threshold a crossover event is flagged through SFR[WTHH].
15:0 THRL	WDG Lower Threshold Value This indicates the lower threshold for the converted data on any channel, below which threshold cross over event is flagged through SFR[WTHL].

30.6.2.11 Latest Data Register (SDADC_LDREG)

Offset 0x0028 Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 313. Latest Data Register (SDADC_LDREG)

Table 356. SDADC_LDREG field descriptions

Field	Description
15:0 LDR	Latest Data Register This field holds the value of the latest converted data word.

30.6.2.12 Latest Time-Stamp Register (SDADC_LTREG)

Offset 0x002C — Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	LTR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 314. Latest Timestamp Register (SDADC_LTREG)

Table 357. SDADC_LTREG field descriptions

Field	Description
23:0 LTR	Latest Time-Stamp Register This field holds the value of the latest timestamp whenever timestamp capture is enabled.

30.7 Functional description

The SDADC has four distinct available modes determined by fields in MCR.

- Differential input mode
- Single-ended input mode
- External modulator mode
- Low consumption mode

30.7.1 Differential input mode

After system reset exit, this is the default mode of operation if SDADC is enabled by asserting MCR[EN]. This mode is entered by negating MCR[MODE]. In differential input mode, a pair of analog inputs among AIN[0:7] is connected to the input terminals of the SDADC modulator block. The module supports four differential pairs: AIN[0,1], AIN[2,3], AIN[4,5], and AIN[6,7].

The modulator block has two input terminals, positive (INP) and negative (INM). The selection of input pair depends on the ANCHSEL field of CSR. The connectivity of each analog input in a given differential pair, whether it is connected to positive or negative terminal, is determined according to *Table 346*.

The differential input mode can also be used to perform gain and offset calibration, where the input terminals are biased with fixed voltages coming from the bias block.

30.7.2 Single-ended input mode

Single-ended input mode is entered by asserting MCR[MODE]. In this mode, one of the analog inputs among AIN[0:7] is selected by the ANCHSEL field of CSR, and is connected to the positive terminal. The negative input terminal is biased with a fixed internal voltage. There are two internal bias voltages available: reference ground (VREFN) and half-scale bias (VREFP/2). The selection of this common mode voltage is controlled by MCR[VCOMSEL].

30.7.3 External modulator mode

This mode is selected by asserting the MCR[EMSEL] field. When external modulator is selected, the internal modulator can be put in low consumption mode by deasserting the MCR[EN] field. In this mode, the data stream and clock outputs from the external modulator available on BSEXT/CLKEXT input pins are multiplexed with internal modulator data output and input sampling clock (f_s). External modulator data must be synchronized with respect to the falling edge of CLKEXT. In this mode, hardware trigger input is ignored even if it is enabled by asserting the MCR[TRIGEN] field. All other configuration settings (such as gain, decimation rate, filter) programmed in MCR have no effect on the conversion.

30.7.4 Low consumption mode

This is the default mode after exiting from reset. This mode is exited by asserting the MCR[EN] field to enable the SDADC block. When data conversion is not required, the SDADC internal modulator can be put in low consumption mode to reduce power consumption.

30.7.5 Analog input multiplexing and bias support

The complete selection and connectivity of analog inputs AIN[0:7] is based on the ANCHSEL field of CSR and the MODE and VCOMSEL bits of MCR, according to *Table 346*. Each analog input can be biased to half scale (VDD_HV_ADV/2) via on-chip 100 KΩ resistance. This allows simple AC coupling to the external analog inputs through a capacitor. This biasing is applicable irrespective of whether analog channel is selected through multiplexer or not.

30.7.6 Programmable gain and decimation rate

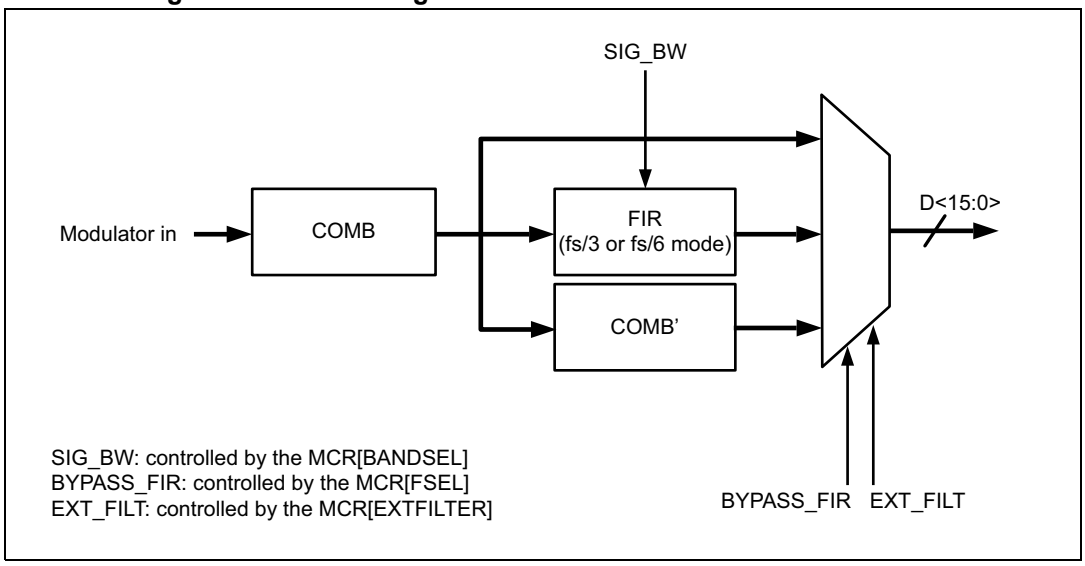
All analog inputs can be configured to have a selectable input gain as defined in the PGAN field description in MCR. This means the input signal is sampled and the result is amplified by the factor determined by PGAN before providing the same to modulator.

The SDADC module is provided with a fixed input sampling clock f_s. The input signal is grossly oversampled by the modulator in order to reduce quantization noise. To support different pass-bands with a fixed sampling clock, a programmable decimation rate is implemented. The process of decimation, to eliminate redundant data at the output while retaining the necessary information, is controlled by the field PDR in MCR.

30.7.7 SD ADC Internal Filter Selection

SD ADC Decimation filter module is composed of COMB and FIR filter added in cascaded manner when default setting is used. COMB filter is always active for the SD ADC data path whereas SD ADC provides several configurable options for the selection of filter after COMB. *Figure 315* illustrates the decimation filter module inside SD ADC.

Figure 315. Block Diagram of SD ADC Decimation Filter Module



Following are the possible valid configurations for SD ADC Decimation Filter Module.

Figure 316. SD ADC Decimation Filter Module with default setting

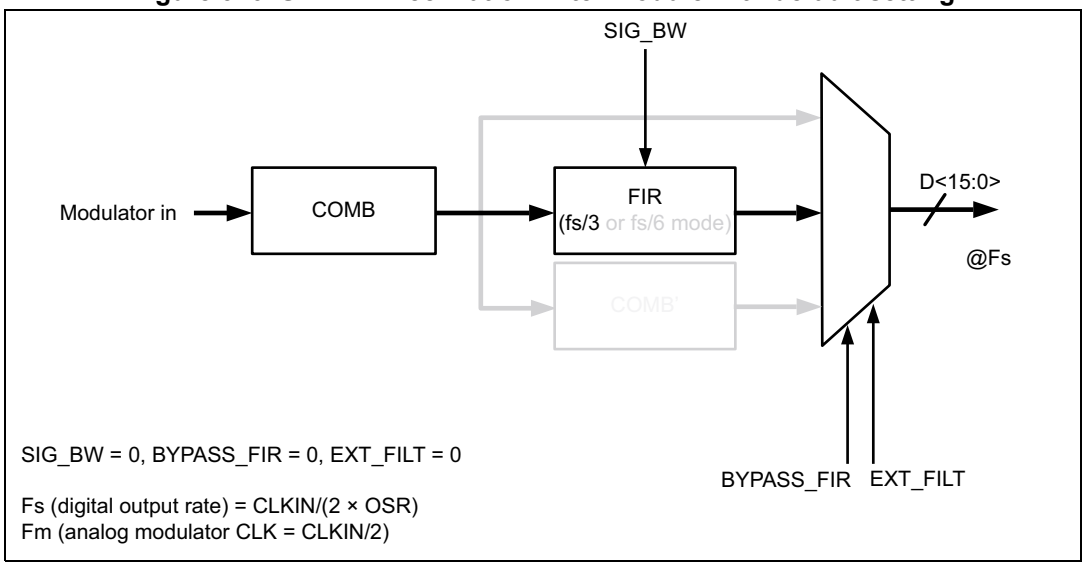


Figure 317. SD ADC Decimation Filter Module with modified bandwidth

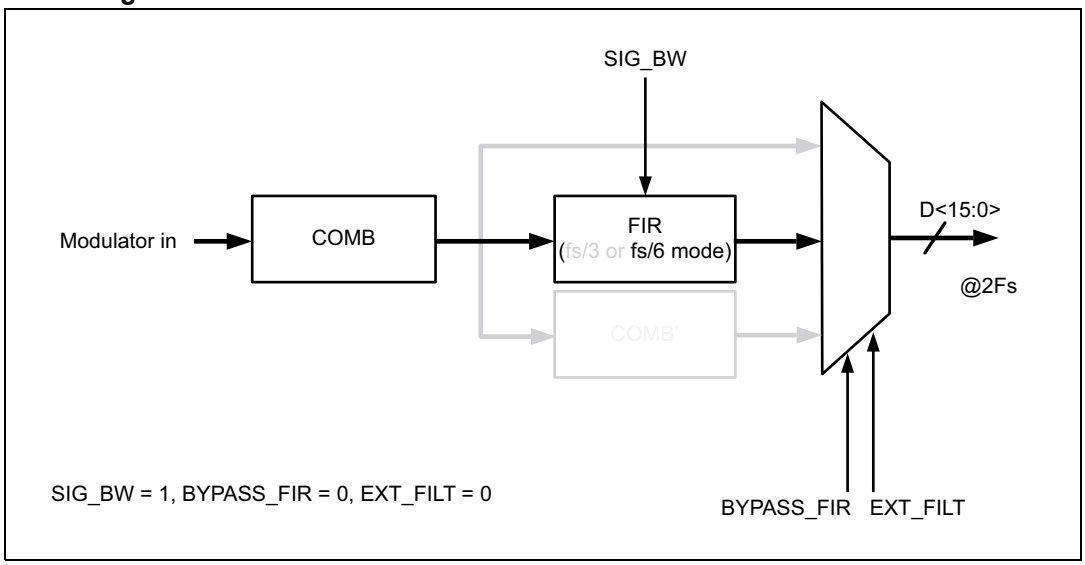


Figure 318. SD ADC Decimation Filter Module with bypass FIR

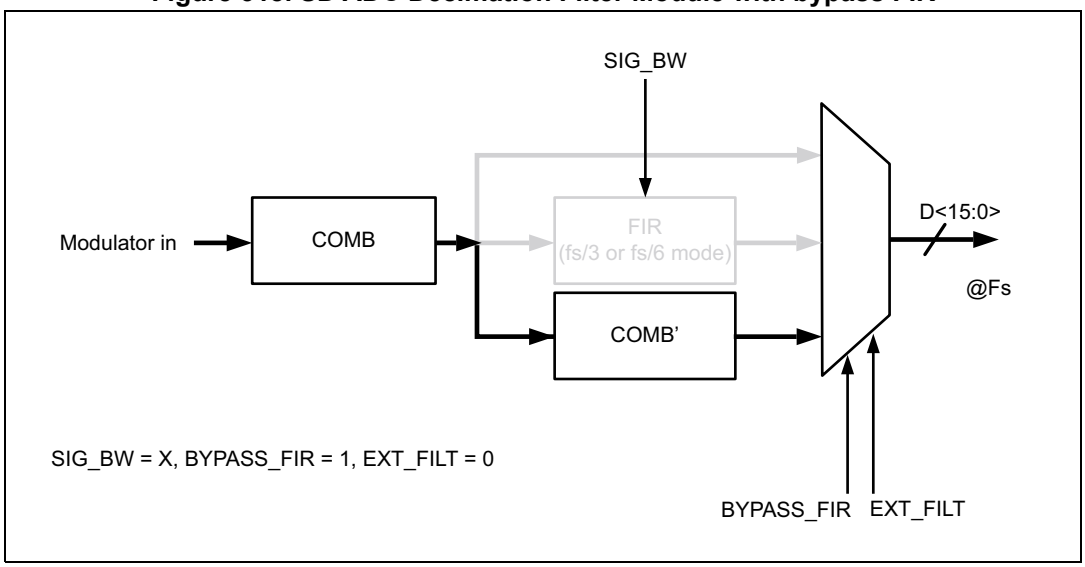
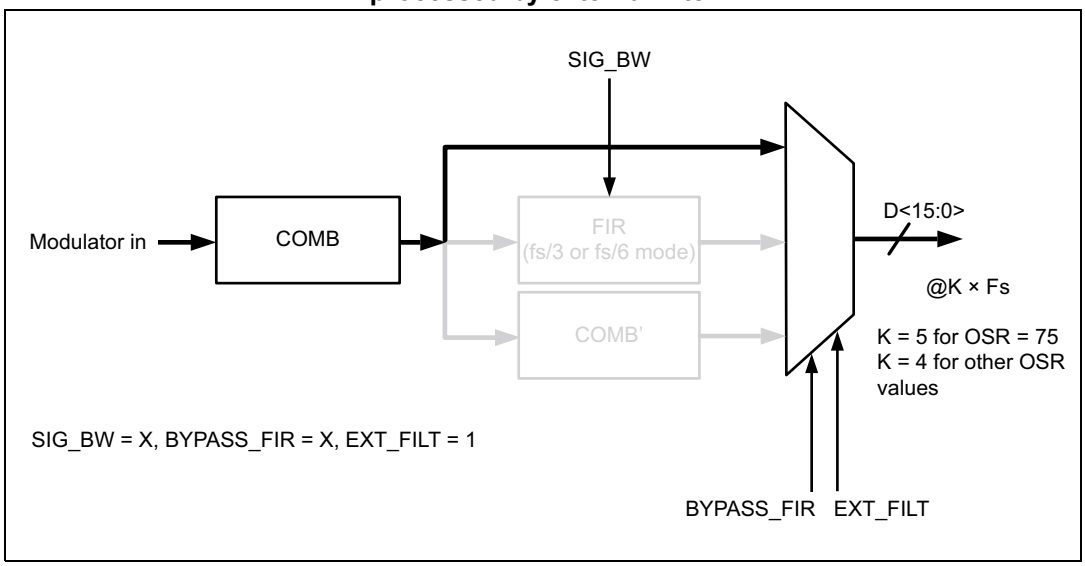


Figure 319. SD ADC Decimation Filter Module with raw data from COMB to be processed by external filter



When SD ADC Decimation Filter Module is configured to provide raw data from COMB filter (shown in *Figure 319*), an external Decimation filter must be used for further decimation.

30.7.8 High-pass filter support

For pure AC applications, it is useful to remove any DC component in the input signal. An optional high-pass filter is implemented which can be enabled by asserting the MCR[HPFEN] field. The –3 dB frequency of the filter is fixed (not programmable) and is available in the device data sheet. This high-pass filter is implemented in the decimation filter logic of the SDADC and is applicable only to internal modulator mode.

30.7.9 Data conversion

Software Control Mechanism: the SDADC block is in continuous conversion mode as soon as it is enabled. The conversion data is not accurate enough for some time, depending on the latency and output settling time specifications. This is true even when there is any change in the configuration settings provided by MCR, change in the analog channel selection due to mux switching time. To account for these delays, an internal timer is implemented which counts down from a start value determined by the OSD field of OSDR. This timer is reloaded whenever the RESET_KEY of RKR is written with 0x5AF0, which is required to start SDADC conversion from a fresh state with a changed configuration. The timer is stalled on reaching count ‘0’ and asserts the SFR[CDVF] field. Once this flag is asserted, converted data is loaded into data FIFO on every rising edge of the output clock (f_d).

A write access to the MCR and CSR registers deasserts the CDVF flag bit to indicate that data coming from SDADC is not valid or corrupted. If the CDVF flag is reset, the flags DFFF and DFORF are blocked from becoming asserted, that is transitioning from inactive state to active state. If DFFF bit is already set, this means the data FIFO is already full before CDVF deassertion.

The DFF flag bit (which is not conditioned by the interrupt or DMA selection) is provided outside which can be routed to timer module at SoC level.

Wraparound Control Mechanism: Here a wrap-around logic must include all analog channels of the input multiplexer from Ain(“000“) to Ain(CSR[ANCHSEL_WRAP]) in both Single ended and Differential ended mode in wraparound sequence triggered by Hardware or software source:

- MCR[WRMODE] bit enables or disables the wraparound mechanism. Clearing the WRMODE bit automatically switches FSM back to Software control mode.
- Every valid trigger advances the wraparound counter to next channel to be converted.
- Wraparound Counter: an internal wraparound counter which points to analog channel to be converted takes the initial entry value for the first loop of execution from CSR[ANCHSEL] while entering wraparound mode. From next loop onwards the counter wraparounds to default initial value “000“. Maximum value of the counter (wraparound value) must be programmed in CSR[ANCHSEL_WRAP] before entering wraparound mode, definition of ANCHSEL and ANCHSEL_WRAP is provided in *Table 346*.
- Upon entering or during wraparound mode user must take care to program ANCHSEL value < ANCHSEL_WRAP value, otherwise hardware would reject the new ANCHSEL value.
- Possible trigger sources:
  - SOURCE1 (Software Bit): writing to STKR.ST_KEY at any time during the wraparound mode (irrespective of MCR[TRIGEN or TRIGSEL] setting) generates a trigger pulse, each such trigger initiates a fresh conversion and advances the wraparound counter by one.

Note: Care must be taken while programming the TRIGEN/TRIGSEL of each SDADC instance that no unintended simultaneous SW triggering of multiple instances happen.

  - SOURCE2 (Hardware Trigger): Here external trigger output enabled by MCR[TRIGEN or TRIGEDSEL or TRIGSEL] before entering wraparound mode for selective or all SDADC instances provides trigger, each such valid trigger initiates a fresh conversion and advances the wraparound counter by one. One of the hw_trig_in ports can be used to connect the external trigger output.
- Next channel number to be converted can be updated by software even in the middle of wraparound sequence execution, this can be done by rewriting CSR[ANCHSEL]. After CSR[ANCHSEL] is rewritten, the next trigger takes updated value (wraparound counter would increment from the updated value).
- Current wraparound counter value can be read from SFR[ANCHSEL_CNT] bit field.

30.7.10 Hardware triggering

Software Control Mechanism: SDADC conversion can be triggered by various hardware events coming from either external pins or internal timers. The field TRIGSEL of MCR selects which hardware event is to be used for triggering conversions. This trigger event, which is synchronous to the peripheral clock, is enabled by asserting MCR[TRIGEN]. As soon as a trigger event is detected, reset input of SDADC block is asserted synchronously for four cycles. This ensures that SDADC starts from a fresh state when reacting to the input trigger event. The SDADC internal modulator must be enabled before enabling the trigger event — otherwise the trigger event is ignored. A hardware trigger event is also ignored when external modulator is selected (MCR[EMSEL] = 1).

If the same trigger event is provided to multiple SDADC digital interface modules and the respective enable controls (TRIGEN bit MCR) are asserted, synchronous operation of multiple SDADC blocks can be achieved with output results being updated at the same time.

Once the reset input is deasserted, the first valid conversion output is stored in data FIFO after a configurable delay determined by OSDR[OSD].

It is also possible to generate a software trigger event output by a write access of STKR with a predefined key. This event output can be used as a trigger to achieve synchronous software start of multiple SDADC blocks.

In case of a hardware trigger, the skew/jitter is less than 100 ns. It is more than one ADC clock cycle due to synchronization. In case of a software trigger, all SDADCs start synchronous sampling.

Wraparound Control Mechanism: in wraparound control mode, hardware and software triggers are equally valid. The hardware trigger source has to be selected before enabling the wraparound mode.

30.7.11 Watchdog

A watchdog threshold crossover event occurs when the SD ADC converted value is greater than watchdog high threshold value or lower than the watchdog low threshold value. This event is indicated by the following two types of signals (also shown in *Figure 321*):

- Edge Triggered Signal - This signal is used to generate DMA/Interrupt request.
- Level Triggered Signal - This signal indicates how long an out-of-range condition is valid (for example, to indicate an overcurrent condition to the timer modules. Refer to the TIMx configuration sections for the interconnection details).

In default mode, the hysteresis is disabled on ADC watchdog upper and lower threshold values. The hysteresis is enabled by programming MCR[LTWM] bit to 1. The behavior with hysteresis enabled is shown in *Figure 320*.

Figure 320. Level triggered signal generated during Watchdog Crossover Event with Hysteresis mode enabled

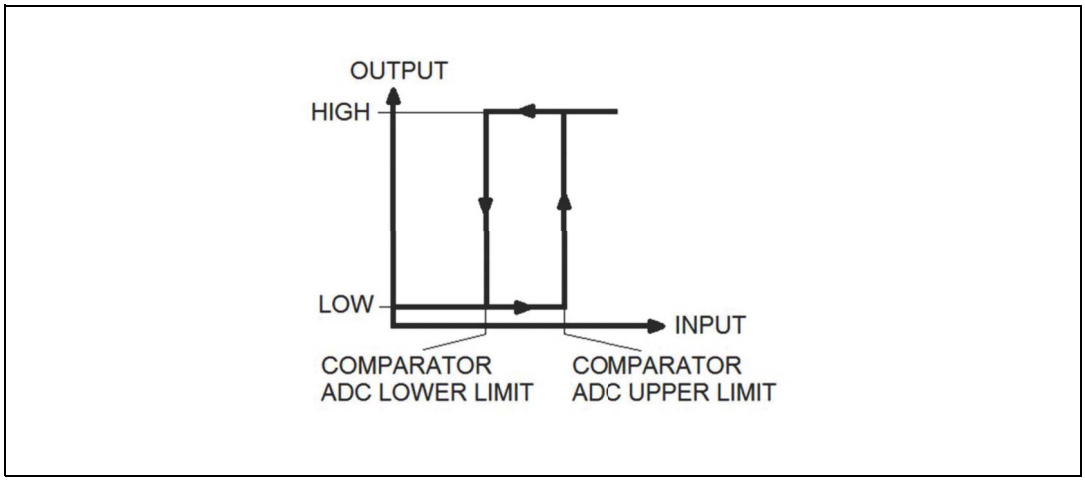
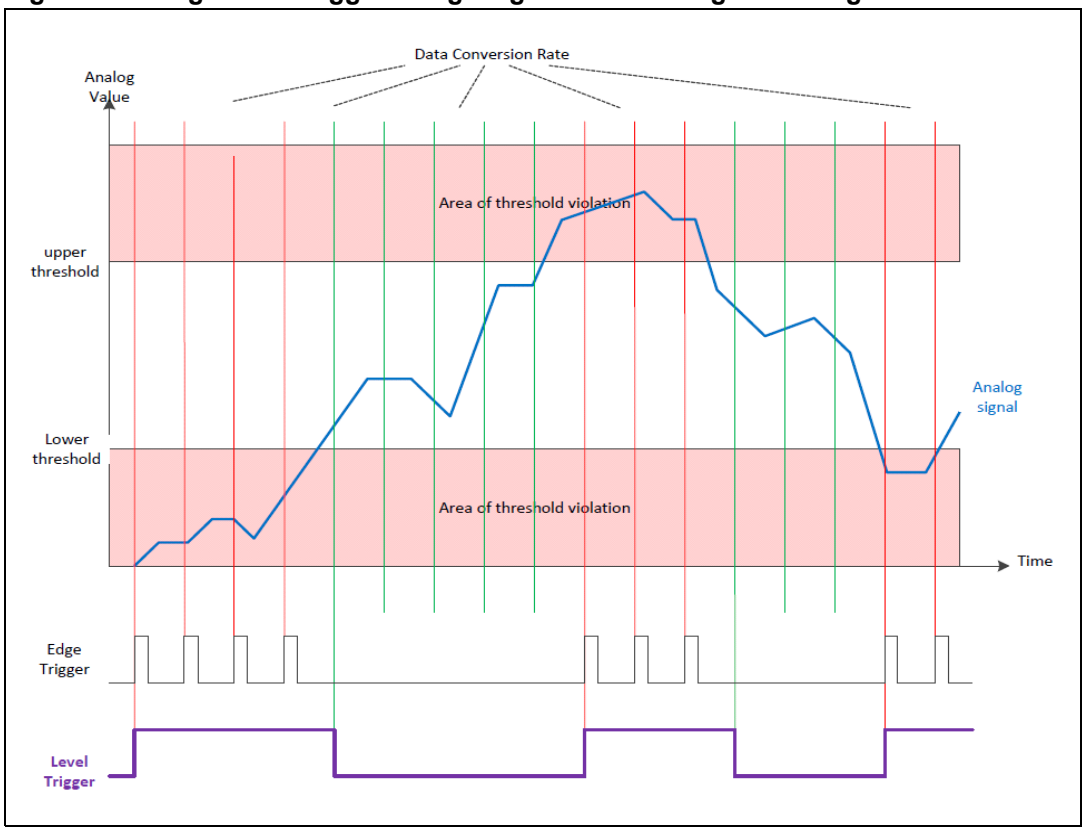


Figure 321. Edge/Level triggered signal generated during Watchdog Crossover Event



30.7.12 Interrupt/DMA request support

The SDADC has one condition that generates interrupts only and another that generates interrupts or DMA request.

The DFFF field is asserted when the data FIFO full condition is detected, indicating that the number of converted data words stored in data FIFO is equal to or greater than the number of data words indicated by FCR[FTHLD] value. The interrupt or DMA request is generated only when RSER[DFFDIRE] is asserted. RSER[DFFDIRS] selects whether a DMA request or an interrupt request is generated.

The data FIFO overrun flag (DFORF) indicates the FIFO overflow condition. Further converted data words cannot be received and data is lost when FIFO overwrite functionality is disabled. The interrupt request is generated only when RSER[DFORIE] is asserted.

Apart from above, a WDG threshold crossover event can also generate an interrupt or DMA request.

If RSER[GGE] is set, all module interrupt/DMA requests are qualified only when an external global gating signal is asserted and RSER[DIRFWGS] bit is '0'.

30.7.13 Gain calibration support

To perform gain calibration, the following sequence needs to be applied.

1. Select differential mode of operation by writing MCR[MODE] to ‘0’.
2. Enable gain error calibration mode by writing MCR[GECEN] to ‘1’.
3. Configure the mux selection ANCHSEL field of CSR to ‘110’. This configuration applies VREFP on positive terminal and VREFN on negative terminal.
4. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 0x00.
5. Disable the high-pass filter by deasserting MCR[HPFEN].
6. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
7. Wait the minimum time equal to 2*(8/Fs), (Fs= SDADC_CLK/2*OSD), for stable output.
8. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. The measurement has to be repeated to reduce contribution of noise during calibration process. D_p is the average value of attenuated positive full scale given by D_p = AVERAGE(CDR[CDATA]).
9. Change the mux selection ANCHSEL field of CSR to ‘111’. This configuration applies VREFN on positive terminal and VREFP on negative terminal.
10. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
11. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. The measurement has to be repeated to reduce contribution of noise during calibration process. Dn is the average value of attenuated negative full scale given by Dn = AVERAGE(CDR[CDATA]).
12. The SDADC calibrated gain can be calculated as:
    $Gain = (D_p – D_n) / 2^{16}$
13. The measured gain value can be used to nullify the gain errors in the digital output. For calibrated conversion, the data CDR[CDATA] provided by the SDADC has to be normalized using the calculated gain: $CDATA_{norm}$ is given by $CDATA_{norm}$ = CDR[CDATA]/Gain.

The calibration calculated by the above procedure is also valid in the case of a single ended configuration.

Note: The gain error calibration has to only be done with gain equal to ‘1’ (in other words, PGAN = 000). Nevertheless, the calculated gain can be applied to normalized data also for conversion with GAIN > 1.

MCR[GECEN] = 1 ensures the accurate gain error mode is enabled. This bit has to be set during both calibration process and after calibration, for application data conversion. Application data conversion must then be normalized using the measured gain.

The higher the number of full-scale conversion (D_p, D_n) done, the higher the rejection of noise during the calibration. The number of conversion is dependent on application noise. It is recommended to run at least 16 conversions when calculating an average value.

Gain calibration has to be done using the same OSR configuration, FIR filter selection mode and output data rate band selection as the target application, since full-scale values may vary slightly with these settings (Normal mode and Bypass FIR mode).

Refer to Digital output codes in full scale table in the SR5E1x microcontroller Datasheet for Full-Scale values (with MCR[GECEN] = 1) with different OSR settings, both for Normal and Bypass FIR modes.

30.7.14 Offset calibration support

To perform offset calibration, the following sequence must be applied:

1. Select differential mode of operation by writing MCR[MODE] to ‘0’.
2. Configure the mux selection ANCHSEL field of CSR to ‘100’ or ‘101’ as required.
   a) CSR = ‘100’ in case of data conversion after calibration in “single ended mode with negative input = $V_{SS_HV_ADR_D}$”
   b) CSR = ‘101’ in case of data conversion after calibration in “differential mode” and “single ended mode with negative input = ($V_{DD_HV_ADR_D}$–$V_{SS_HV_ADR_D}$)/2”
3. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 0x00.
4. Disable the high pass filter by deasserting MCR[HPFEN].
5. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
6. Read the digital output stored in FIFO after the output settling time.
7. The measured offset can be used to nullify the offset error in the digital output. Expected output is 0b00_0000_0000_0000. The SDADC offset can be calculated as:

   Offset = Expected Output – Actual Output

Note: The offset must be calculated for each PGAN field setting since it is expected to vary with gain configuration of SDADC.

30.7.15 Global gating

The generation of specific events is gated based upon an external gating signal when Global Gating Feature is enabled. The feature is enabled by asserting Request Select and Enable Register RSER [GGE] bit.

The following sequence lists the events supported when this feature is enabled:

1. Gating DMA/Interrupt request
2. Gating FIFO write operation
3. Gating communication between SD-ADC and the companion module (Decimation filter) through PSI

30.7.15.1 Gating DMA/Interrupt request

This is the default gating event selected when Global Gating feature is enabled. It also allows an external gate signal to be synchronized with new ADC sample. The synchronized external gate signal is selected by asserting Request Select and Enable Register RSER [SGARE] bit.

30.7.15.2 Gating FIFO write operation

The operation to write new ADC sample to FIFO is decided based on an external gate signal when Global Gating feature is enabled. This feature is selected by asserting Request Select and Enable Register RSER [DIRFWGS]. Based on RSER [CLERE] the stored last event is also copied inside FIFO when an external gate is open. The last event is captured in Latest Data Register (LDREG).

It is also captured in Latest Timestamp Register (LTREG) when timestamp feature is enabled (RSER [CTRE] bit is '1').

Note: FIFO Threshold has to be set to "0000" (1 Data word) while gating FIFO write operation is used. It allows reading of all the data words stored inside FIFO.

30.7.15.3 Gating communication between SD-ADC and Decimation filter through PSI

The transfer of every new ADC sample through PSI to Decimation filter is decided based on an external gate signal. It is applicable only when PSI feature is enabled (PSICR [PSIEN] bit is '1') and Global Gating feature is enabled (RSER [GGE] bit is '1').

30.7.16 Timestamp

The SDADC module allows external timestamp to be stored inside FIFO along with converted data. The timestamp feature is enabled by asserting Request Select and Enable Register (RSER) CTRE bit.

The timestamp is captured at the moment when new ADC sample is present. FIFO first stores the sampled data, and the corresponding timestamp at the next location. Whenever timestamp feature is enabled, the sampled data and its timestamp are considered as 1 data word though timestamp is stored at the next location. It allows FIFO Control Register (FCR) FCR [FTHLD] and FCR [FSIZE] bit field interpretation to be same. Hence, if software programs FCR [FTHLD] to “1111”, FIFO FULL condition is generated once FIFO is filled with 16 data words whereas each data word is composed of data and timestamp stored at consecutive location.

The READ interface of FIFO is through Converted Data Register (CDR) only. Hence, in case of interrupt mode, software must read this register in multiples of two. First read always gives the data and the second read provides the corresponding timestamp. When DMA mode is used, linked DMA feature is used to read data followed by timestamp.

The converted data is 16 bits wide whereas timestamp is 24 bits wide. Hence, for the first read 16 bits are valid and for the second read 24 bits are valid of Converted Data Register (CDR).

Once the converted data is present, the data and timestamp get stored at Latest Data Register (LDREG) and Latest Timestamp Register (LTREG) respectively. The storage of LDREG and LTREG registers is independent of gating functionality. The registers get overwritten each time a new data arrives.

30.8 Initialization information

To initialize the SDADC registers for data conversion, the following sequence is required.

1. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
2. Enable high-pass filter if required.
3. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
4. Configure OSD delay according to SDADC startup time or latency from reset exit.
5. Enable the SDADC by asserting MCR[EN].
6. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR. Otherwise, if data conversion need to be triggered by a hardware event, assert MCR[TRIGEN].
7. DFFF interrupt or DMA request is generated after the data FIFO has reached threshold to indicate that data can be transferred.

The following sequence needs to be followed when external modulator mode of SDADC is selected for data conversions.

1. Disable the SDADC internal modulator by deasserting MCR[EN].
2. Configure MCR to select the external modulator mode.
3. Select the required external pins BSEXT/CLKEXT in the GPIO configuration of the device.
4. Configure OSD delay according to external modulator startup time or latency from reset exit.
5. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
6. DFFF interrupt or DMA request is generated after the data FIFO has reached threshold to indicate that data can be transferred.

30.8.1 Data conversion step

To acquire a data from SDADC, the following sequence is required:

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter is required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to SDADC required startup time or latency from reset exit.
6. Generate a reset event writing 0x5AF0 in RESET_KEY field of RKR register.
7. Wait till FIFO empty flag DFEF of SFR register is clear.
8. Read data by CDATA of CDR register.

Repeat steps from 6 to 8 for new acquisitions.

Note: The time elapsed between reset event and Read data by CDATA field of CDR register is:

- *Reset event <- -> Data Valid flag (CDVS) = Output settling Time delay (OSDR has to be set at least to 16)*
- *Data Valid flag <- -> Data FIFO is not empty = 0.5*CLK_Out + 3*CLK_In*
- *Data FIFO is not empty <- -> read Data (safety point) = 2*CLK_In*

*where: CLK_Out = CLK_In/(2*OSR)*

31 Digital-to-analog converter (DAC)

31.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC features up to two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, V_{REF+} (shared with other analog peripherals) is available for better resolution.

The DAC_OUTx pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on-chip peripheral. The DAC output buffer can be optionally enabled to allow a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support sample and hold mode.

31.2 DAC main features

The DAC main features are the following (see *Figure 322: Dual-channel DAC block diagram*)

- Up to five DAC interfaces, maximum two output channels each
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Sawtooth wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DAC_OUTx output pin
- DAC output connection to on chip peripherals
- Input voltage reference, V_{REF+}

Figure 322 shows the block diagram of a DAC channel and *Table 359* gives the pin description.

31.3 DAC implementation

Table 358. DAC implementation

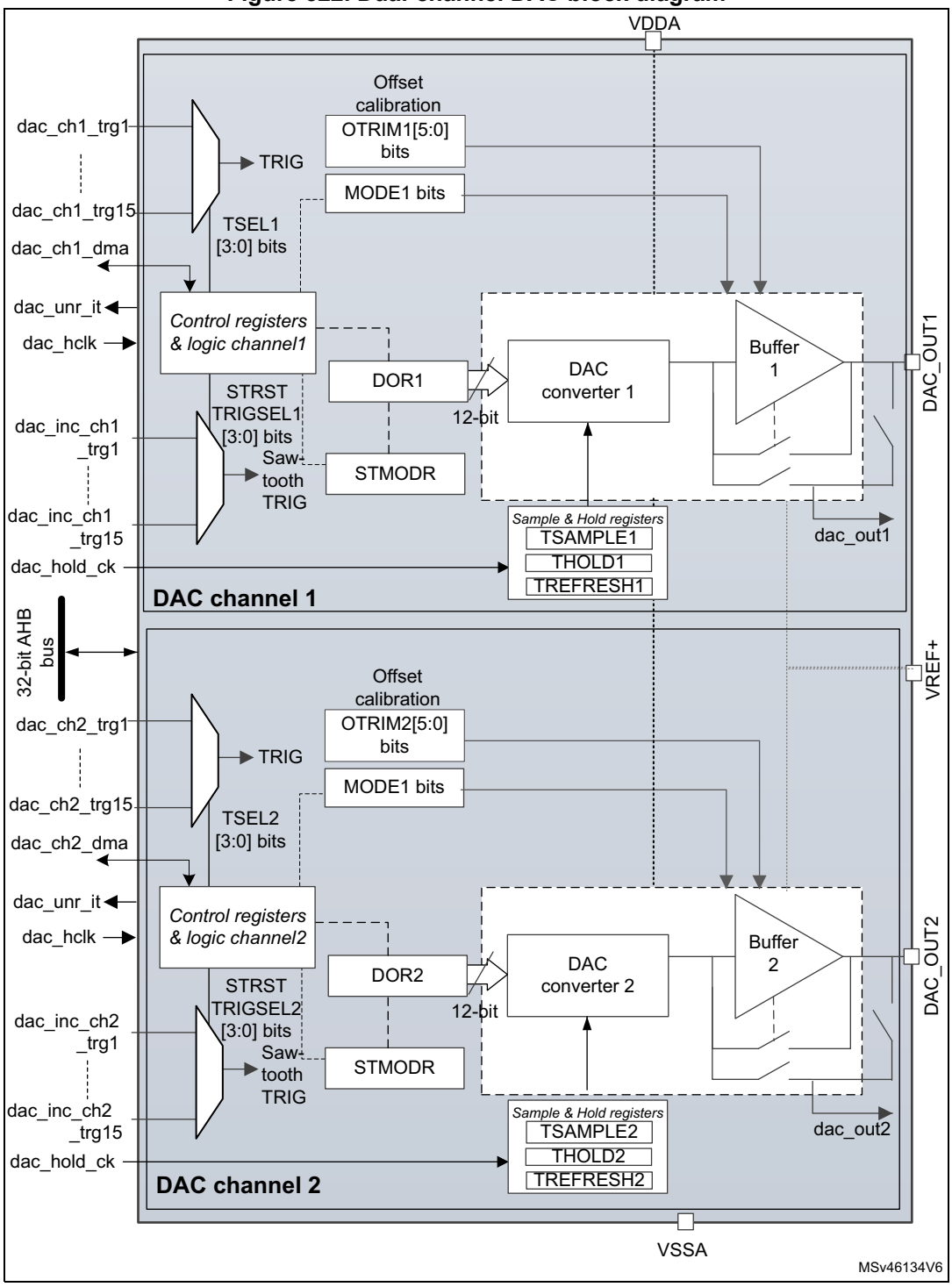
DAC features	B-DAC	DAC1 to DAC4
Dual channel	X	X
Output buffer	X	-
I/O connection	B-DAC_OUT1 on PB1 B-DAC_OUT2 on PC5	No connection to a GPIO
Maximum sampling time(1)	1MSPS	15MSPS

1. Refer to the device datasheet for sampling time.

31.4 DAC functional description

31.4.1 DAC block diagram

Figure 322. Dual-channel DAC block diagram



Note: MODEx bits in the DAC_MCR control the output mode and allow switching between the normal mode in buffer/unbuffered configuration and the sample and hold mode.

Refer to Section 31.3: DAC implementation.

31.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DAC_OUTx can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DAC_OUTx output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not.

Table 359. DAC input/output pins

Pin name	Signal type	Remarks
VREF+	Input, analog reference positive	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

Table 360. DAC input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 15)	Inputs	DAC channel2 trigger inputs
dac_ch1_inc_trgx (x = 1 to 15)	Inputs	DAC channel1 sawtooth increment trigger inputs
dac_chn2_inc_trgx (x = 1 to 15)	Inputs	DAC channel1 sawtooth increment trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_hclk	Input	DAC peripheral clock
dac_hold_ck	Input	DAC low-power clock used in Sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals
dac_out2	Analog output	DAC channel2 output for on-chip peripherals

Table 361. BDAC1 and DAC1 interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi	LSI/32 clock enabled by RCC
dac_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	EXTI9	External pin
dac_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_chx_trg9 (x = 1, 2)	hrtim1_dac_reset_trg_A	Internal signal from on-chip timers
dac_chx_trg10 (x = 1, 2)	hrtim1_dac_reset_trg_B	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	hrtim1_dac_reset_trg_C	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	hrtim1_dac_reset_trg_D	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	hrtim1_dac_reset_trg_E	Internal signal from on-chip timers
dac_chx_trg14 (x = 1, 2)	hrtim1_dac_reset_trg_F	Internal signal from on-chip timers
dac_chx_trg15 (x = 1, 2)	hrtim1_dac_trg1	Internal signal from on-chip timers
dac_inc_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg6 (x = 1, 2)	EXTI10	External pin
dac_inc_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg9 (x = 1, 2)	hrtim1_dac_step_trg_A	Internal signal from on-chip timers
dac_inc_chx_trg10 (x = 1, 2)	hrtim1_dac_step_trg_B	Internal signal from on-chip timers
dac_inc_chx_trg11 (x = 1, 2)	hrtim1_dac_step_trg_C	Internal signal from on-chip timers
dac_inc_chx_trg12 (x = 1, 2)	hrtim1_dac_step_trg_D	Internal signal from on-chip timers
dac_inc_chx_trg13 (x = 1, 2)	hrtim1_dac_step_trg_E	Internal signal from on-chip timers
dac_inc_chx_trg14 (x = 1, 2)	hrtim1_dac_step_trg_F	Internal signal from on-chip timers

Table 362. DAC2 interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi	LSI/32 clock enabled by RCC
dac_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	EXTI9	External pin
dac_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_chx_trg9 (x = 1, 2)	hrtim1_dac_reset_trg_A	Internal signal from on-chip timers
dac_chx_trg10 (x = 1, 2)	hrtim1_dac_reset_trg_B	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	hrtim1_dac_reset_trg_C	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	hrtim1_dac_reset_trg_D	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	hrtim1_dac_reset_trg_E	Internal signal from on-chip timers
dac_chx_trg14 (x = 1, 2)	hrtim1_dac_reset_trg_F	Internal signal from on-chip timers
dac_chx_trg15 (x = 1, 2)	hrtim1_dac_trg2	Internal signal from on-chip timers
dac_inc_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg6 (x = 1, 2)	EXTI10	External pin
dac_inc_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg9 (x = 1, 2)	hrtim1_dac_step_trg_A	Internal signal from on-chip timers
dac_inc_chx_trg10 (x = 1, 2)	hrtim1_dac_step_trg_B	Internal signal from on-chip timers
dac_inc_chx_trg11 (x = 1, 2)	hrtim1_dac_step_trg_C	Internal signal from on-chip timers
dac_inc_chx_trg12 (x = 1, 2)	hrtim1_dac_step_trg_D	Internal signal from on-chip timers
dac_inc_chx_trg13 (x = 1, 2)	hrtim1_dac_step_trg_E	Internal signal from on-chip timers
dac_inc_chx_trg14 (x = 1, 2)	hrtim1_dac_step_trg_F	Internal signal from on-chip timers

Table 363. DAC3 interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi	LSI/32 clock enabled by RCC
dac_chx_trg1 (x = 1, 2)	TIM1_TRGO	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	EXTI9	External pin
dac_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_chx_trg9 (x = 1, 2)	hrtim2_dac_reset_trg_A	Internal signal from on-chip timers
dac_chx_trg10 (x = 1, 2)	hrtim2_dac_reset_trg_B	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	hrtim2_dac_reset_trg_C	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	hrtim2_dac_reset_trg_D	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	hrtim2_dac_reset_trg_E	Internal signal from on-chip timers
dac_chx_trg14 (x = 1, 2)	hrtim2_dac_reset_trg_F	Internal signal from on-chip timers
dac_chx_trg15 (x = 1, 2)	hrtim2_dac_trg1	Internal signal from on-chip timers
dac_inc_chx_trg1 (x = 1, 2)	TIM1_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg6 (x = 1, 2)	EXTI10	External pin
dac_inc_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg9 (x = 1, 2)	hrtim2_dac_step_trg_A	Internal signal from on-chip timers
dac_inc_chx_trg10 (x = 1, 2)	hrtim2_dac_step_trg_B	Internal signal from on-chip timers
dac_inc_chx_trg11 (x = 1, 2)	hrtim2_dac_step_trg_C	Internal signal from on-chip timers
dac_inc_chx_trg12 (x = 1, 2)	hrtim2_dac_step_trg_D	Internal signal from on-chip timers
dac_inc_chx_trg13 (x = 1, 2)	hrtim2_dac_step_trg_E	Internal signal from on-chip timers
dac_inc_chx_trg14 (x = 1, 2)	hrtim2_dac_step_trg_F	Internal signal from on-chip timers

Table 364. DAC4 interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi	LSI/32 clock enabled by RCC
dac_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	EXTI9	External pin
dac_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_chx_trg9 (x = 1, 2)	hrtim2_dac_reset_trg_A	Internal signal from on-chip timers
dac_chx_trg10 (x = 1, 2)	hrtim2_dac_reset_trg_B	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	hrtim2_dac_reset_trg_C	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	hrtim2_dac_reset_trg_D	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	hrtim2_dac_reset_trg_E	Internal signal from on-chip timers
dac_chx_trg14 (x = 1, 2)	hrtim2_dac_reset_trg_F	Internal signal from on-chip timers
dac_chx_trg15 (x = 1, 2)	hrtim2_dac_trg1	Internal signal from on-chip timers
dac_inc_chx_trg1 (x = 1, 2)	TIM8_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg2 (x = 1, 2)	TIM7_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg3 (x = 1, 2)	TIM15_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg4 (x = 1, 2)	TIM2_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg5 (x = 1, 2)	TIM4_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg6 (x = 1, 2)	EXTI10	External pin
dac_inc_chx_trg7 (x = 1, 2)	TIM6_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg8 (x = 1, 2)	TIM3_TRGO	Internal signal from on-chip timers
dac_inc_chx_trg9 (x = 1, 2)	hrtim2_dac_step_trg_A	Internal signal from on-chip timers
dac_inc_chx_trg10 (x = 1, 2)	hrtim2_dac_step_trg_B	Internal signal from on-chip timers
dac_inc_chx_trg11 (x = 1, 2)	hrtim2_dac_step_trg_C	Internal signal from on-chip timers
dac_inc_chx_trg12 (x = 1, 2)	hrtim2_dac_step_trg_D	Internal signal from on-chip timers
dac_inc_chx_trg13 (x = 1, 2)	hrtim2_dac_step_trg_E	Internal signal from on-chip timers
dac_inc_chx_trg14 (x = 1, 2)	hrtim2_dac_step_trg_F	Internal signal from on-chip timers

31.4.3 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

31.4.4 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

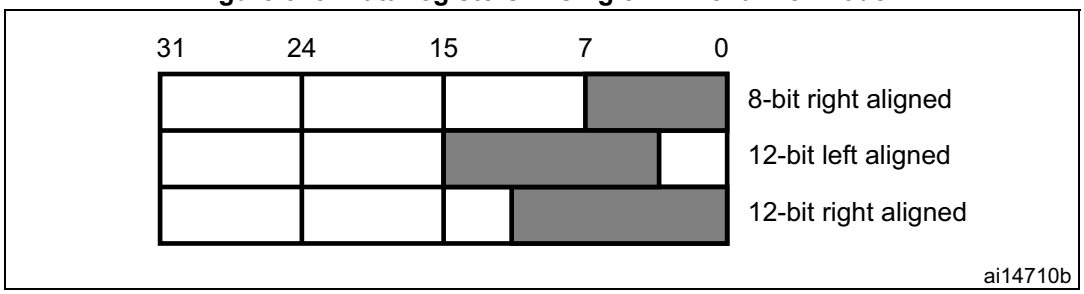
  – 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

Figure 323. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

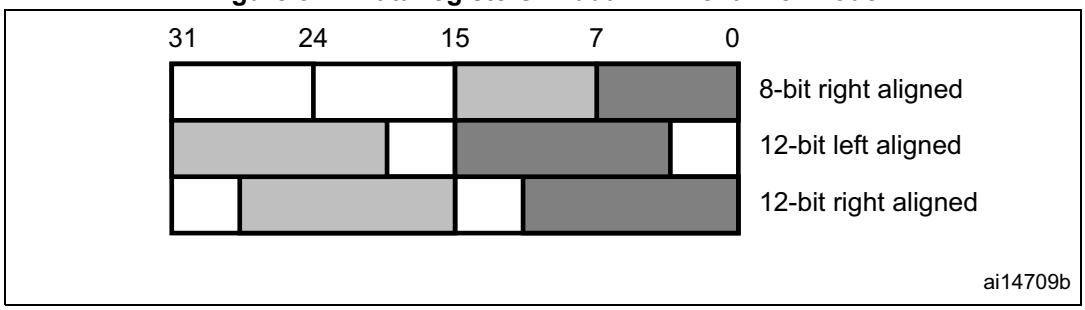
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

  – 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 324. Data registers in dual DAC channel mode



Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2’s complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB bit of the data written to DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB bit is inverted. The four LSBs are simply ignored.

Table 365. Data format (case of 12-bit data)

SINFORMATx bit	DATA written to DHRx register	DATA transferred to DORx register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

31.4.5 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_hclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_hclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

HFSEL bits of DAC_MCR must be set when dac_hclk clock speed is faster than 80 MHz. It adds an extra delay to the transfer from DAC_DHRx register to DAC_DORx register.

Refer to the table *HFSEL description* below for the limitation of the DAC_DORx update rate depending on HFSEL bits and dac_hclk clock frequency.

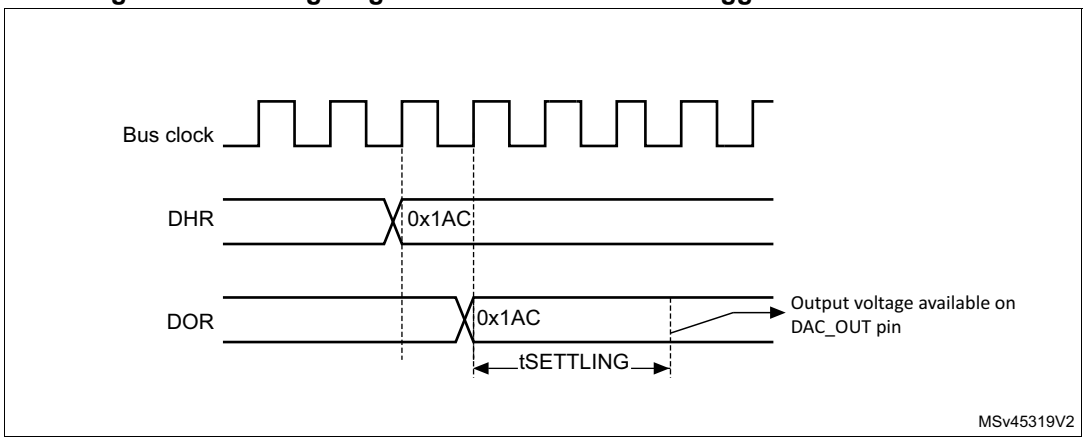
If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

The above timing is only related to the limitation of the DAC interface. Refer also to the $t_{SETTLING}$ parameter value in the product datasheet.

Table 366. HFSEL description

HFSEL[1:0]	AHB frequency	Function
00	⩽ 80 MHz	DAC_DOR update rate up to 3 AHB clock cycles
01	>80 MHz	DAC_DOR update rate up to 5 AHB clock cycles
10	>160 MHz	DAC_DOR update rate up to 7 AHB clock cycles
11	Reserved	—

Figure 325. Timing diagram for conversion with trigger disabled TEN = 0



31.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACoutput} = V_{REF} \times \frac{DOR}{4096}$$

31.4.7 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in *Section 31.4.2: DAC pins and internal signals*.

Each time a DAC interface detects a rising edge on the selected trigger source, the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_hclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

The reset trigger selection and the increment trigger selection of the sawtooth generation are performed through STRSTTRIGSELx and STINCTRIGSELx control bits, respectively. STRSTTRIGSELx mapping is similar to TSELx. Refer to *Section 31.4.2: DAC pins and internal signals* for TSELx, STRSTTRIGSELx, and STINCTRIGSELx mappings.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_hclk clock cycle.

31.4.8 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

DMA Double data mode

When the DMA controller is used in Normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.
2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA Double data mode.

In DMA Double mode, DMA requests can only handle one DAC channel. To use two channel outputs in DMA Double mode, each DMA channel has to be configured separately.

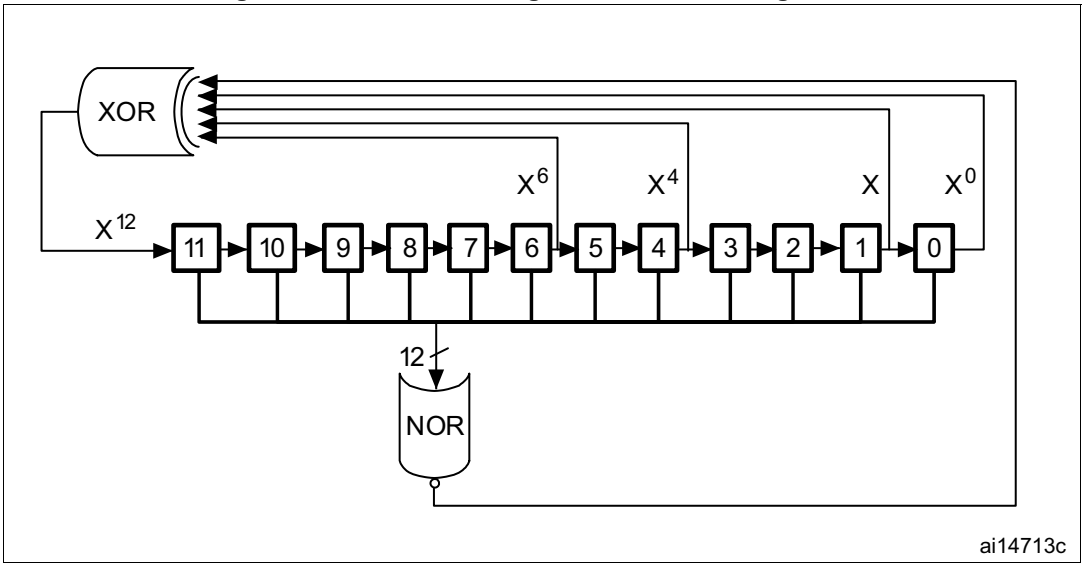
The following conditions must be met to change from Double data to single data mode or vice versa:

- The DAC must be disabled.
- DMAEN bit must be cleared (ENx=0 and DMAENx=0).

31.4.9 DAC noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_hclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 326. DAC LFSR register calculation algorithm

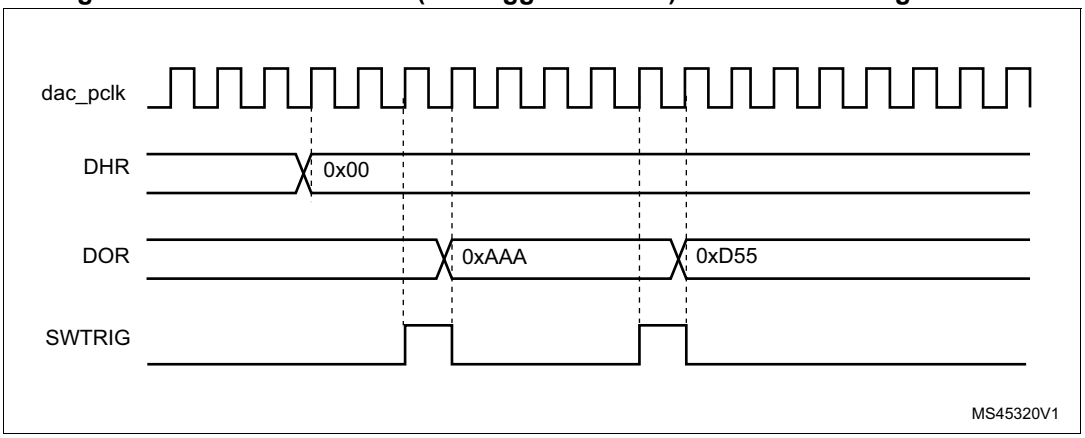


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a ‘1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 327. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

31.4.10 DAC triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_hclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 328. DAC triangle wave generation

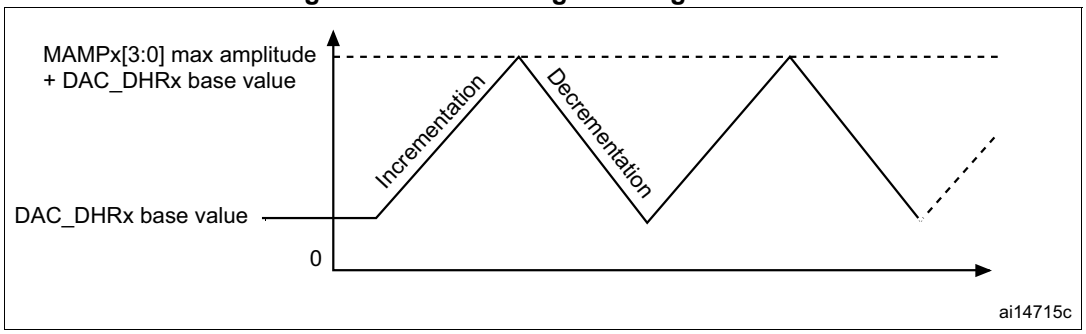
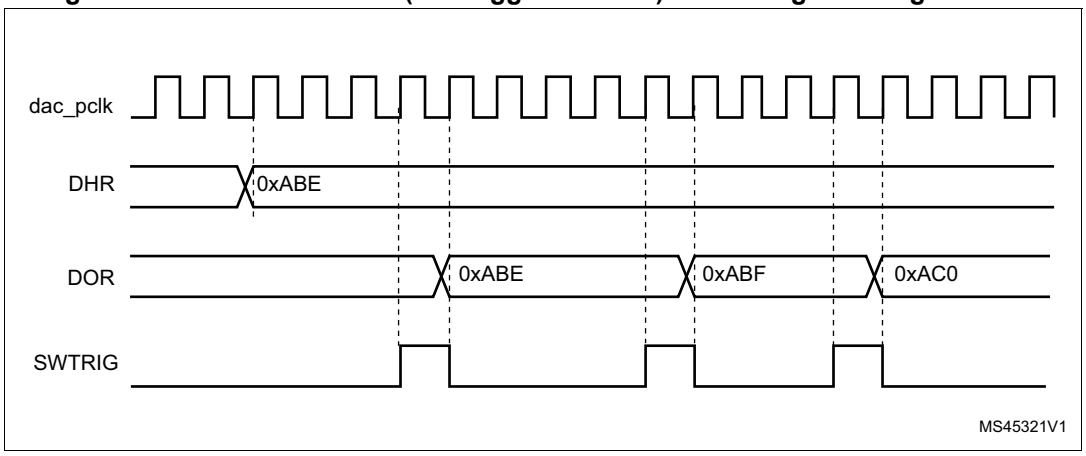


Figure 329. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

31.4.11 DAC sawtooth wave generation

The DAC can generate a sawtooth waveform. Specific register settings for the initial value, increment value and direction control are required:

- DAC sawtooth wave generation is selected by setting WAVEx[1:0] to 11 in the DAC_CR register.
- The sawtooth counter initial value (reset value) is configured through STRSTDATAx[11:0] bits in the DAC_STRx register.
- The increment value is defined by the STINCDATAx[15:0] bits in the DAC_STRx register.
- The sawtooth direction is defined by STDIRx bit in the DAC_STRx register.

The sawtooth counter starts from STRSTDATAx[11:0] (bits 12 to 15 are set to 0000), each increment trigger then increments (or decrements) STINCDATAx[15:0] value.

The DAC output is used from 12 MSB of those counter value. When the counter reaches 0x0000 or 0xFFFF, the value is saturated. The sawtooth reset trigger signal initializes the counter value to the STRSTDATAx[11:0] (bits 12 to 15 are set to 0000) value.

The increment trigger and reset trigger must be selected through the STINCTRIGSELx[3:0] and the STRSTTRIGSELx[3:0] bits.

Figure 330. DAC sawtooth wave generation (STDIRx=0)

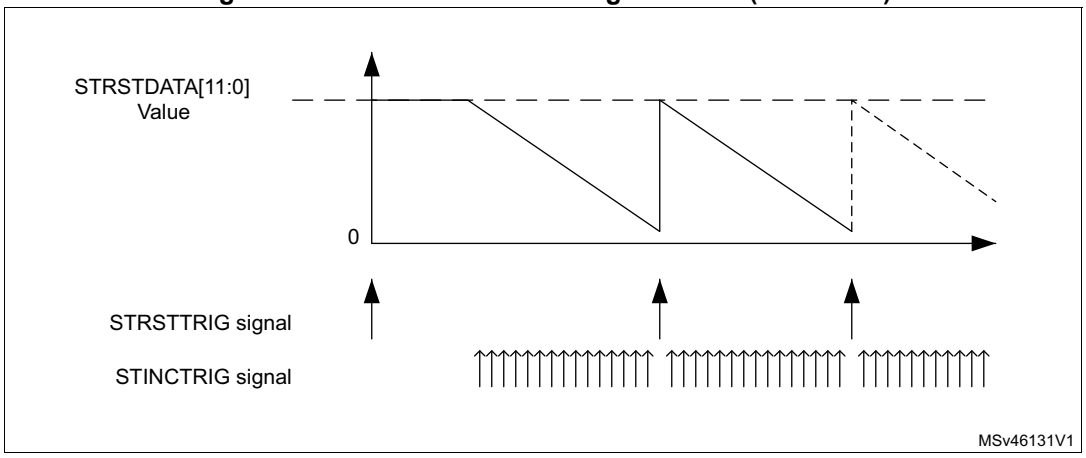
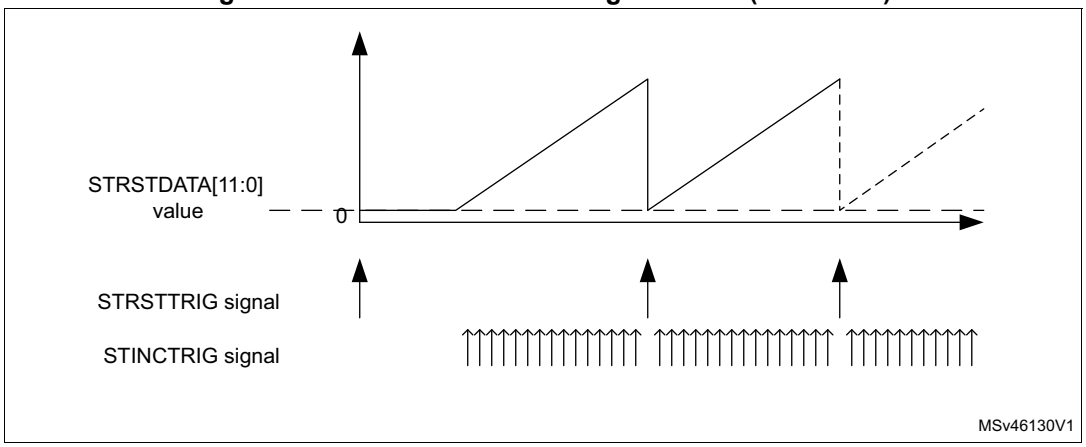
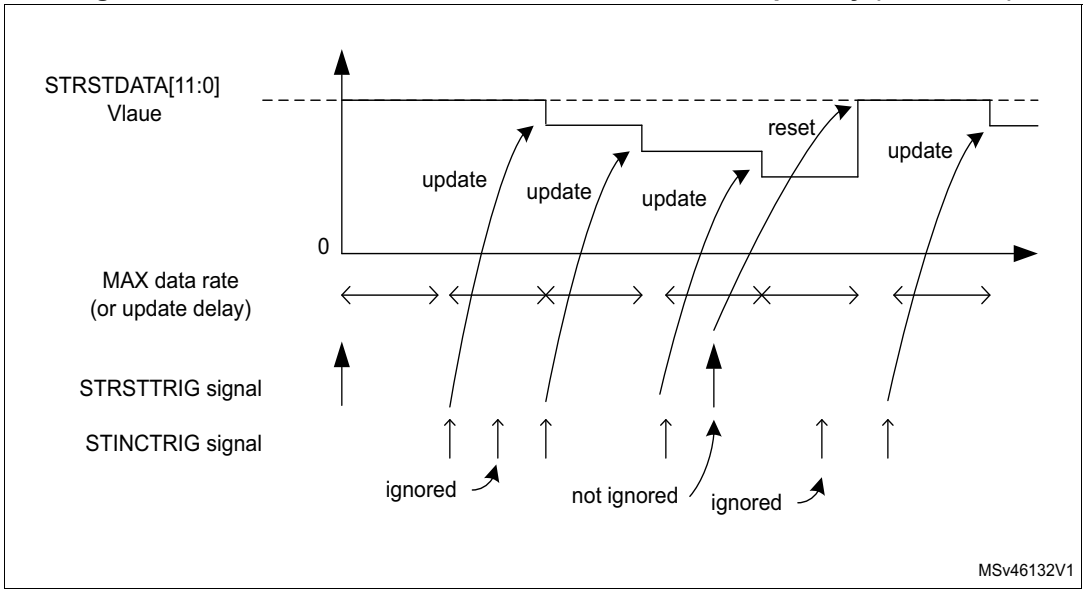


Figure 331. DAC sawtooth wave generation (STDIRx=1)



The STRSTTRIG signal has higher priority than STINCTRIG. The trigger signal cannot be faster than the DAC_DORx update rate defined in the *Table 366: HFSEL description*. If STINCTRIG is asserted faster than the allowed data update rate, the STINCTRIG trigger is ignored. If the STRSTTRIG signal is applied after the STINCTRIG and before DAC_DORx update rate constraints, STRSTTRIG is put on hold. Then, immediately after the data increment, the reset trigger is applied.

Figure 332. DAC sawtooth STINCTRIG and STRSTTRIG priority (STDIR = 0)



31.4.12 DAC channel modes

Each DAC channel can be configured in normal mode or sample and hold mode. The output buffer can be enabled to allow a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In normal mode, there are four combinations, by changing the buffer state and by changing the DAC_OUTx pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, whose value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI low-speed clock (dac_hold_ck) in addition to the dac_hclk clock.

The LSI low-speed clock (dac_hold_ck) must not be stopped on-the-fly when the Sample and hold mode is enabled.

The sample/hold mode operations can be divided into 3 phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI ~32 KHz is selected:

12 cycles are required for sample phase: TSAMPLEx[9:0] = 11,

62 cycles are required for hold phase: THOLDx[9:0] = 62,

and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus Normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 367. Sample and refresh timings

Buffer State	t_{SAMP} (1)(2)	$t_{REFRESH}$ (2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in Sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

C_{SH} = 100 nF

V_{DDA}= 3.0 V

Sampling phase:

t_{SAMP} = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms
(where R_{BON} = 2 kΩ)

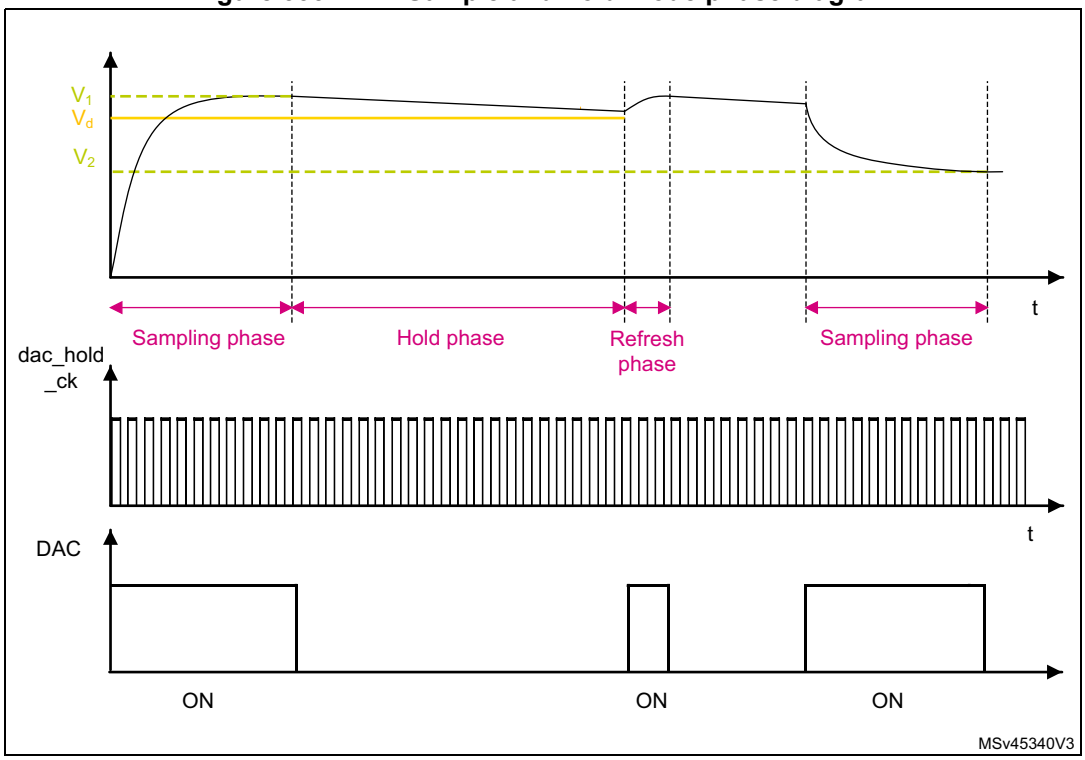
Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs
(where N_{LSB} = 10 (10 LSB drop during the hold phase)

Hold phase:

D_v = i_{leak} * t_{hold} / C_{SH} = 0.0073 V (10 LSB of 12bit at 3 V)
i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)
t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 333. DAC Sample and hold mode phase diagram



Like in Normal mode, the Sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

  When MODEx[2:0] bits are equal to 111, an internal capacitor, C_{Lint}, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All Sample and hold phases are interruptible, and any change in DAC_DHRx immediately triggers a new sample phase.

Note: The sawtooth wave generation is not supported in Sample and hold mode operation.

Table 368. Channel output modes summary

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (such as comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (such as comparators)
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

31.4.13 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = ((D/2^{N-1}) \times G \times V_{ref}) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{ref} is the nominal full-scale voltage, and V_{os} is the offset voltage. For an ideal DAC channel, G = 1 and V_{os} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{os}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 000b or 001b or 100b or 101b). If applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to VREF+/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.

- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, VREF+ values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the Nominal factory trimming conditions

In addition, when V_{DD} is removed the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 000b or 001b or 100b or 101b.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 00000b.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) If CAL_FLAGx is set to 1, the OTRIMx[4:0] trimming code is found and can be used during *device* operation to compensate the output value, else increment OTRIMx[4:0] and repeat sub-steps from (a) to (d) again.

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

The commutation/toggle of CAL_FLAGx bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIMx[4:0] bits in DAC_CCR register.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

When CENx bit is set, it is not allowed to set ENx bit.

31.4.14 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

15 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bit fields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_hclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_hclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bit fields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bit fields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bit fields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure different trigger sources by setting different values in STRSTTRIGSEL1[3:0], STRSTTRIGSEL2[3:0], STINCTRIGSEL2[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx values for each register.

When a DAC channel1 trigger arrives, the DAC channel1 sawtooth counter updates the DHR1 register and transfers it into DAC_DOR1 (three AHB clock cycles later).

When a DAC channel2 trigger arrives, the DAC channel2 sawtooth counter updates the DHR1 register and transfers it into DAC_DOR1 (three AHB clock cycles later).

Independent trigger with different sawtooth wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure different trigger sources by setting different values in the STRSTTRIGSEL1[23:0], STRSTTRIGSEL2[23:0], STINCTRIGSEL2[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits as 11 and set different STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx value for each register.

When a DAC channel1 trigger arrives, the DAC channel1 sawtooth counter updates the DHR1 register and loads it into DAC_DOR1 (three AHB clock cycles later).

When a DAC channel2 trigger arrives, the DAC channel2 sawtooth counter updates the DHR2 register and loads it into DAC_DOR1 (three AHB clock cycles later).

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_hclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bit fields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_hclk clock cycles).

Simultaneous trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bit fields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask,

is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bit fields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bit fields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bit fields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into

DAC_DOR1 (three AHB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the DAC channel2 triangle counter is updated.

Simultaneous trigger with single sawtooth wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the same trigger source for both DAC channels by setting the same value in STRSTTRIGSEL1[3:0],STRSTTRIGSEL2[3:0], STINCTRIGSEL2[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx value for each register.

When a trigger arrives, the DAC channel1/2 sawtooth counter updates DHR1 and DHR2 registers and loads them into DAC_DOR1/2 (three AHB clock cycles later).

Simultaneous trigger with different sawtooth wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the same trigger source for both DAC channels by setting the same value in STRSTTRIGSEL1[3:0], STRSTTRIGSEL2[3:0], STINCTRIGSEL2[3:0] bits and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits to 11 and set different STRSTDATAx[11:0], STINCDATAx[15:0], STDIRx values for each register.

When a trigger arrives, DAC channel1/2 sawtooth counter updates the DHR1 and DHR2 registers and loads them independently into DAC_DOR1/2 (three AHB clock cycles later).

31.5 DAC low-power modes

Table 369. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA.

31.6 DAC interrupts

Table 370. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode
DAC	DMA underrun	DMAUDRx	DMAUDRIEx	Write DMAUDRx = 1	Yes

31.7 DAC registers

Refer to *Section 1.2: Register conventions* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

31.7.1 DAC memory map

Table 371. DAC register memory map

Offset	Register name
0x00	*DAC control register (DAC_CR)*
0x04	*DAC software trigger register (DAC_SWTRGR)*
0x08	*DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)*
0x0C	*DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)*
0x10	*DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)*
0x14	*DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)*
0x18	*DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)*
0x1C	*DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)*
0x20	*Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)*
0x24	*Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)*
0x28	*Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)*
0x2C	*DAC channel1 data output register (DAC_DOR1)*
0x30	*DAC channel2 data output register (DAC_DOR2)*
0x34	*DAC status register (DAC_SR)*
0x38	*DAC calibration control register (DAC_CCR)*
0x3C	*DAC mode control register (DAC_MCR)*
0x40	*DAC channel1 sample and hold sample time register (DAC_SHSR1)*
0x44	*DAC channel2 sample and hold sample time register (DAC_SHSR2)*
0x48	*DAC sample and hold time register (DAC_SHHR)*
0x4C	*DAC sample and hold refresh time register (DAC_SHRR)*
0x58	*DAC channel1 sawtooth register (DAC_STR1)*
0x5C	*DAC channel2 sawtooth register (DAC_STR2)*
0x60	*DAC sawtooth mode register (DAC_STMODR)*

31.7.2 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3]	TSEL2[2]	TSEL2[1]	TSEL2[0]	TEN2	EN2
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3]	TSEL1[2]	TSEL1[1]	TSEL1[0]	TEN1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in Normal operating mode

1: DAC channel2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Note: These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Sawtooth wave generation enabled

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)
These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

0000: SWTRIG2
0001: dac_ch2_trig1
0010: dac_ch2_trig2
...
1111: dac_ch2_trig15

Refer to the trigger selection tables in *Section 31.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_hclk clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_hclk clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_hclk clock cycle.
These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

0: DAC channel2 disabled

1: DAC channel2 enabled

Note: These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1=0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in Normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: wave generation disabled

01: Noise wave generation enabled

10: Triangle wave generation enabled

11: Sawtooth wave generation enabled

Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trig1
0010: dac_ch1_trig2
...
1111: dac_ch1_trig15
Refer to the trigger selection tables in *Section 31.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.
Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable
This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_hclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_hclk clock cycles later to the DAC_DOR1 register
Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_hclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable
This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

31.7.3 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIGB2	SWTRIGB1
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **SWTRIGB2**: DAC channel2 software trigger B
This bit is set by software to trigger the DAC in software trigger mode (sawtooth generation) It is cleared by hardware.
0: No trigger
1: Trigger for sawtooth increment
Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 16 **SWTRIGB1**: DAC channel1 software trigger B
This bit is set by software to trigger the DAC in software trigger mode (sawtooth generation) It is cleared by hardware.
0: No trigger
1: Trigger for sawtooth increment

Bit 15:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

31.7.4 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in Double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

31.7.5 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC1DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC1DHRB[11:0]**: DAC channel1 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in Double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

31.7.6 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHRB[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC1DHRB[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in Double data mode.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

31.7.7 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHRB[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in DMA Double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.

31.7.8 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHRB[11:0]**: DAC channel2 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in Double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

31.7.9 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHRB[7:0]								DACC2DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHRB[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel2 when the DAC operates in Double data mode.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

31.7.10 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

31.7.11 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

31.7.12 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

31.7.13 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DORB[11:0]**: DAC channel1 data output
These bits are read-only. They contain data output for DAC channel1 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

31.7.14 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DORB[11:0]**: DAC channel2 data output
These bits are read-only. They contain data output for DAC channel2 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.

31.7.15 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											

Bit 31 **BWST2**: DAC channel2 busy writing sample time flag

This bit is systematically set just after Sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 30 **CAL_FLAG2**: DAC channel2 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 28 **DORSTAT2**: DAC channel2 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in Double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 27 **DAC2RDY**: DAC channel2 ready status bit

This bit is set and cleared by hardware.

0: DAC channel2 is not yet ready to accept the trigger nor output data

1: DAC channel2 is ready to accept the trigger or output data

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after Sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 **DORSTAT1**: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in Double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Bit 11 **DAC1RDY**: DAC channel1 ready status bit

This bit is set and cleared by hardware.

0: DAC channel1 is not yet ready to accept the trigger nor output data

1: DAC channel1 is ready to accept the trigger or output data

Bits 10:0 Reserved, must be kept at reset value.

31.7.16 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

31.7.17 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
						rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFSEL1	HFSEL0	Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
rw	rw					rw	rw						rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SINFORMAT2**: Enable signed format for DAC channel2

This bit is set and cleared by software.

0: Input data is in unsigned format

1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 24 **DMADOUBLE2**: DAC channel2 DMA double data mode

This bit is set and cleared by software.

0: DMA Normal mode selected

1: DMA Double data mode selected

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE2[2:0]:** DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2=0 and bit CEN2 =0 in the DAC_CR register). If EN2=1 or CEN2 =1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

– DAC channel2 in Normal mode

000: DAC channel2 is connected to external pin with Buffer enabled

001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled

010: DAC channel2 is connected to external pin with buffer disabled

011: DAC channel2 is connected to on chip peripherals with Buffer disabled

– DAC channel2 in Sample and hold mode

100: DAC channel2 is connected to external pin with Buffer enabled

101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel2 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN2=0.

Refer to Section 31.3: DAC implementation for the availability of DAC channel2.

Bits 15:14 **HFSEL[1:0]**: High frequency interface mode selection
00: High frequency interface mode disabled
01: High frequency interface mode compatible to AHB>80 MHz enabled
10: High frequency interface mode compatible to AHB>160 MHz enabled
11: Reserved

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1
This bit is set and cleared by software.
0: Input data is in unsigned format
1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Bit 8 **DMADOUBLE1**: DAC channel1 DMA double data mode
This bit is set and cleared by software.
0: DMA Normal mode selected
1: DMA Double data mode selected

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1=0 and bit CEN1 =0 in the DAC_CR register). If EN1=1 or CEN1 =1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
– DAC channel1 in Normal mode
000: DAC channel1 is connected to external pin with Buffer enabled
001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
010: DAC channel1 is connected to external pin with Buffer disabled
011: DAC channel1 is connected to on chip peripherals with Buffer disabled
– DAC channel1 in sample & hold mode
100: DAC channel1 is connected to external pin with Buffer enabled
101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
111: DAC channel1 is connected to on chip peripherals with Buffer disabled
Note: This register can be modified only when EN1=0.

31.7.18 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in Sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1=1, the write operation is ignored.

*Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI clock period.*

31.7.19 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE2[9:0]:** DAC channel2 sample time (only valid in Sample and hold mode)
These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2=1, the write operation is ignored.

*Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI clock period.*

31.7.20 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **THOLD2[9:0]:** DAC channel2 hold time (only valid in Sample and hold mode).
Hold time= (THOLD[9:0]) x LSI clock period

Note: This register can be modified only when EN2=0.
These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in Sample and hold mode)
Hold time= (THOLD[9:0]) x LSI clock period

Note: This register can be modified only when EN1=0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx=0 and bit CENx=0 in the DAC_CR register). If ENx=1 or CENx=1 the write operation is ignored.

31.7.21 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TREFRESH2[7:0]:** DAC channel2 refresh time (only valid in Sample and hold mode)
Refresh time= (TREFRESH[7:0]) x LSI clock period

Note: This register can be modified only when EN2=0.
These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in Sample and hold mode)
Refresh time= (TREFRESH[7:0]) x LSI clock period

Note: This register can be modified only when EN1=0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx=0 and bit CENx=0 in the DAC_CR register). If ENx=1 or CENx=1 the write operation is ignored.

31.7.22 DAC channel1 sawtooth register (DAC_STR1)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STINCDATA1[15:0]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	STDIR1	STRSTDATA1[11:0]											
			rw	rw											

Bits 31:16 **STINCDATA1[15:0]:** DAC channel1 sawtooth increment value (12.4 bit format)

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **STDIR1:** DAC channel1 sawtooth direction setting
This bit is written by software to select the direction of Sawtooth step direction
0: Decrement
1: Increment

Bits 11:0 **STRSTDATA1[11:0]:** DAC channel1 sawtooth reset value

31.7.23 DAC channel2 sawtooth register (DAC_STR2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STINCDATA2[15:0]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	STDIR2	STRSTDATA2[11:0]											
			rw	rw											

Bits 31:16 **STINCDATA2[15:0]:** DAC channel2 Sawtooth increment value (12.4 bit format)

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **STDIR2:** DAC channel2 sawtooth direction setting
This bit is written by software to select the direction of sawtooth step direction
0: Decrement
1: Increment

Bits 11:0 **STRSTDATA2[11:0]:** DAC channel2 sawtooth reset value

31.7.24 DAC sawtooth mode register (DAC_STMODR)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STINCTRIGSEL2[3:0]				Res.	Res.	Res.	Res.	STRSTTRIGSEL2[3:0]			
				rw								rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	STINCTRIGSEL1[3:0]				Res.	Res.	Res.	Res.	STRSTTRIGSEL1[3:0]			
				rw								rw			

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **STINCTRIGSEL2[3:0]:** DAC channel2 sawtooth increment trigger selection
Refer to the trigger selection tables in *Section 31.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.
0000: SWTRIGB2
0001: dac_inc_ch2_trg1
....
1111: dac_inc_ch2_trg15

Note: These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 19:16 **STRSTTRIGSEL2[3:0]:** DAC channel2 sawtooth reset trigger selection
Refer to the trigger selection tables in *Section 31.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.
0000: SWTRIGB2
0001: dac_ch2_trg1
....
1111: dac_ch2_trg15
The mapping is the same as for TSEL2[3:0].

Note: These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **STINCTRIGSEL1[3:0]:** DAC channel1 sawtooth increment trigger selection
Refer to the trigger selection tables in *Section 31.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.
0000: SWTRIGB1
0001: dac_inc_ch1_trg1
....
1111: dac_inc_ch1_trg15

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **STRSTTRIGSEL1[3:0]:** DAC channel1 sawtooth reset trigger selection
Refer to the trigger selection tables in *Section 31.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.
0000: SWTRIGB1
0001: dac_ch1_trg1
....
1111: dac_ch1_trg15
The mapping is the same as for TSEL1[3:0].

32 Comparator (COMP)

32.1 COMP introduction

The system contains eight analog comparators COMP1 to COMP8 controlled by two digital blocks, COMP_Dig1 and COMP_Dig2.

COMP_Dig1 controls COMP1 to COMP4, and COMP_Dig2 controls COMP5 to COMP8.

The comparators can be used for a variety of functions including:

- Wake-up from low-power mode triggered by an analog signal,
- Analog signal conditioning,
- Cycle-by-cycle current control loop when combined with a PWM output from a timer.

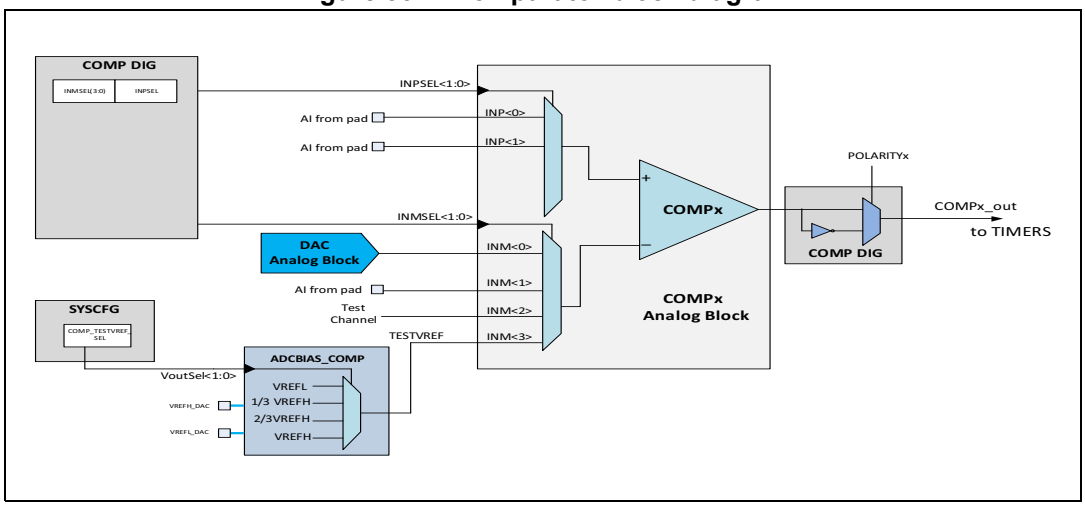
32.2 COMP main features

- Each comparator has configurable plus and minus inputs used for flexible voltage selection.
- INP channels:
  - from up two Multiplexed I/O pins
- INM channels:
  - from one I/O pin
  - from a DAC channel
  - Internal reference voltages (VREFH and VREFL) and two submultiple values (1/3 and 2/3 VREFH) provided by a scaler (buffered voltage divider)
- Programmable hysteresis
- Output redirection to I/Os or to timer inputs for triggering break events for fast PWM shutdowns
- Output blanking for immunity to switching noise
- Per-channel interrupt generation with wake-up from Sleep mode

32.2.1 COMP block diagram

The block diagram of one comparator channel front-end is shown in *Figure 334: Comparator block diagram*.

Figure 334. Comparator block diagram



32.2.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The comparator output can be connected to the I/Os using the alternate function channel given in “Alternate function mapping” table in the SR5E1x pinout Microsoft Excel® file attached to the IO_Definition document.

The output can also be internally redirected to a variety of timer inputs for the following purposes:

- Emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- Cycle-by-cycle current control, using OCREF_CLR inputs
- Input capture for timing measures

It is possible to have the comparator output simultaneously redirected internally and externally.

Table 372. COMPx non-inverting input assignment – INPSEL

INPSEL	COMP1_ INP	COMP2_ INP	COMP3_ INP	COMP4_ INP	COMP5_ INP	COMP6_ INP	COMP7_ INP	COMP8_ INP
0	PB2 (SAR1_IN1)	PB2 (SAR1_IN1)	PB7 (SAR2_IN1)	PB7 (SAR2_IN1)	PB12 (SAR3_IN1)	PB12 (SAR3_IN1)	PC3 (SAR4_IN1)	PC3 (SAR4_IN1)
1	PB3 (SAR1_IN2)	PB4 (SAR1_IN3)	PB8 (SAR2_IN2)	PB9 (SAR2_IN3)	PB13 (SAR3_IN2)	PB14 (SAR3_IN3)	PC4 (SAR4_IN2)	PC6 (SAR4_IN3)

Table 373. COMPx inverting input assignment – INMSEL

INMSEL [3:0]	COMP1_ INM	COMP2_ INM	COMP3_ INM	COMP4_ INM	COMP5_ INM	COMP6_ INM	COMP7_ INM	COMP8_ INM
0000	DAC1_CH1	DAC1_CH2	DAC2_CH1	DAC2_CH2	DAC3_CH1	DAC3_CH2	DAC4_CH1	DAC4_CH2
0001	PB1	PB5 (SAR1_IN4)	PB13 (SAR3_IN2)	PB10 (SAR2_IN4)	PC4 (SAR4_IN2)	PB15 (SAR3_IN4)	PB8 (SAR2_IN2)	PC7 (SAR4_IN4)
0010	Test purpose only							
0011	V_{REFL} - 1/3 V_{REFH} - 2/3 V_{REFH} - V_{REFH}							
0100-1111	Reserved							

Table 374. COMPx – OUT

OUT(1)	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	COMP8
AFSEL=8	PA4	PA5	PA6	PA7	PB1	PD7	PD8	PD9
	PC5	PD11	PD12	PD13	PD14	PD15	PE0	PE2
	PD10	PE3	PE4	PE8	PE9	PE10	PF2	PF3

1. COMPx output OUT selection is performed through dedicated GPIO alternate function selection register (AFSEL).

32.2.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the AHB clock.

There is one COMP-dedicated clock enable control bit in the RCC controller.

Caution: The polarity selection logic and the output redirection to the port work independently of the APB clock.

32.2.4 COMP LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

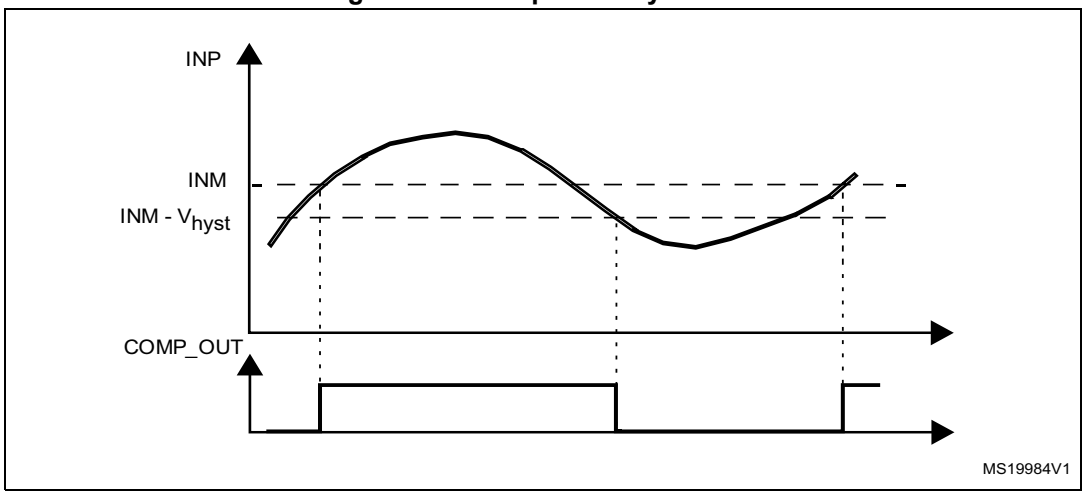
Once the programming is completed, the COMPx LOCK bit can be set. This causes the whole register to become read-only, including the COMPx LOCK bit.

The write protection can only be removed by an MCU reset.

32.2.5 COMP hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions with noisy input signals. It is non-symmetrical and only acting to falling edge of the comparator output. The internal hysteresis function can be disabled so as to set the amount of hysteresis with external components, which can be useful for example when exiting a low-power mode.

Figure 335. Comparator hysteresis



32.2.6 COMP output blanking

The purpose of the blanking function is to prevent the current regulation from tripping upon short current spikes at the beginning of PWM period (typically the recovery current in power switch anti-parallel diodes). This goes through setting a dead window defined with a timer output compare signal. The blanking source is selected individually per comparator channel by software through BLANKSEL[2:0] bitfield of corresponding COMP_CxCSR register, as shown in *Table 375: Blanking sources*. The inverted blanking signal is logical AND-ed with the comparator stage output to produce the comparator channel x output. See the example provided in the following figure.

Figure 336. Comparator output blanking

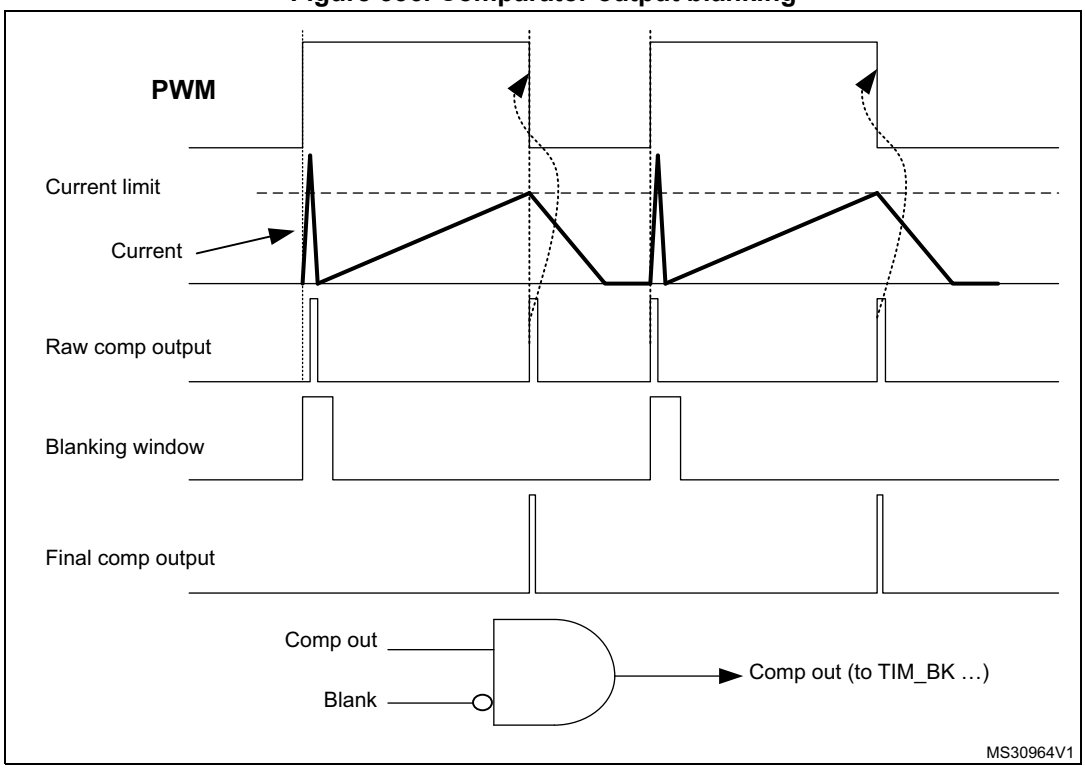


Table 375. Blanking sources

BLANKSEL [2:0]	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	COMP8
0000	Output blanking disabled							
0001	TIM1_OC5	TIM1_OC5	TIM1_OC5	TIM1_OC5	TIM2_OC3	TIM2_OC3	TIM1_OC5	TIM1_OC5
0010	TIM2_OC3	TIM2_OC4	TIM3_OC3	TIM3_OC3	TIM8_OC5	TIM8_OC5	TIM8_OC5	TIM8_OC5
0011	TIM3_OC3	TIM3_OC3	TIM2_OC4	TIM2_OC4	TIM3_OC3	TIM3_OC3	TIM3_OC3	TIM3_OC3
0100	TIM8_OC5	TIM8_OC5	TIM8_OC5	TIM8_OC5	TIM1_OC5	TIM1_OC5	TIM15_OC2	TIM15_OC2
0101	TIM3_OC4	TIM3_OC4	TIM3_OC4	TIM3_OC4	TIM3_OC4	TIM3_OC4	TIM3_OC4	TIM3_OC4
0110	TIM15_OC1	TIM15_OC1	TIM15_OC1	TIM15_OC1	TIM15_OC1	TIM15_OC1	TIM15_OC1	TIM15_OC1
0111	TIM4_OC3	TIM4_OC3	TIM4_OC3	TIM4_OC3	TIM4_OC3	TIM4_OC3	TIM4_OC3	TIM4_OC3

32.3 COMP low-power modes

Table 376. Comparator behavior in low-power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Low-power run	No effect.
Low-power sleep	No effect. COMP interrupts cause the device to exit the Low-power sleep mode.

32.4 COMP interrupts

The comparator outputs are internally connected to the Extended interrupts and events controller. Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit Sleep low-power mode.

Refer to Interrupt and events section for more details.

To enable the COMPx interrupt, it is required to follow this sequence:

1. Configure and enable the EXTI line corresponding to the COMPx output event in interrupt mode and select sensitivity to rising edge, falling edge or to both edges
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines
3. Enable COMPx

Interrupt events are flagged through COMP_CxCSR flags.

32.5 COMP_Dig registers

32.5.1 COMP register map

Table 377. COMP register memory map

Offset	Register name
4(x-1), where x = 1 to 4	*Comparator x control and status register (COMP_CxCSR)*

32.5.2 Comparator x control and status register (COMP_CxCSR)

One COMP_Dig allows to control 4 analog COMP. This section describes COMP_Dig registers.

For x = 1 through 4, the COMP_CxCSR register contains all bits and flags related to the comparator x.

Address offset: 4(x-1), where x = 1 to 4

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.								BLANKSEL[2:0]			HYST		
rw	r									rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL	Res.						INPSEL	INMSEL[3:0]				Res.		DIG_H	EN
rw							rw	rw	rw	rw	rw			rw	rw

Bit 31 **LOCK:** COMP_CxCSR register lock

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator x control register COMP_CxCSR[31:0]. When locked, all control bits and flags can be read only but not written. When unlocked, the control bits can also be written by software.

0: Unlock

1: Lock

Bit 30 **VALUE:** Comparator x output status

This read-only flag reflects the level of the comparator x output before the polarity selector and blanking, as indicated in *Figure 334*.

Bits 29:22 Reserved, must be kept at reset value

Bits 21:19 **BLANKSEL[2:0]:** Comparator x blanking signal select

This bitfield controlled by software selects the blanking signal for comparator channel x, as shown in *Table 375: Blanking sources*.

Bits 18:16 **HYST[2:0]:** Comparator x hysteresis
This bitfield controlled by software selects the hysteresis of the comparator x:
000: No hysteresis
001: 10mV hysteresis
010: 20mv hysteresis
011: 30mV hysteresis
100: 40mV hysteresis
101: 50mV hysteresis
110: 60mV hysteresis
111: 70mV hysteresis

Bit 15 **POL:** Comparator x polarity
This bit controlled by software selects the comparator x output polarity:
0: Non-inverted
1: Inverted

Bits 14:9 Reserved, must be kept at reset value

Bit 8 **INPSEL**: Comparator x signal select for non-inverting input
This bitfield controlled by software selects the signal for the non-inverting input COMPx_INP of the comparator x, as shown in *Table 372: COMPx non-inverting input assignment — INPSEL*.

Bits 7:4 **INMSEL[3:0]**: Comparator x signal select for inverting input
This bitfield controlled by software selects the signal for the inverting input COMPx_INM of the comparator x, as shown in *Table 373: COMPx inverting input assignment — INMSEL*.

Bits 3:1 Reserved, must be kept at reset value

Bit 1 **DIG_H:** COMP2 DIG_H. This bit must be written and cleared by the software.
0x0: The deglitecher is disabled
0x1: The deglitcher is enabled

Bit 0 **EN:** Comparator x enable
This bit controlled by software enables the operation of comparator x:
0: Disable
1: Enable

33 Temperature sensor

33.1 Introduction

The device includes an onboard temperature sensor that monitors device temperature and delivers two analog output signals and three digital output signals.

The analog outputs consist of two voltage signals which vary linearly with the internal junction temperature: a voltage signal that is linearly increasing (PTAT: proportional to absolute temperature) and a voltage signal that is linearly decreasing (CTAT: complementary proportional to absolute temperature). The analog outputs are connected to an input channel of an ADC on the device. The internal junction temperature must be calculated by software based on the converted temperature values.

The three digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions. These signals notify the device to take action to appropriately adjust the device temperature in response to an out of specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

33.1.1 Signal description

This module has no external signals.

33.2 Functional description

The temperature sensor generates two analog output voltages, PTAT and CTAT which are proportional to the absolute current junction temperature of the device.

An on-chip ADC module is used to convert the analog output voltages, PTAT and CTAT, into a digital representation. These values, along with parameter values stored in onboard flash memory, are used by software to calculate the device junction temperature.

33.2.1 Linear temperature sensor (analog output generation)

The temperature sensor outputs two voltages proportional and complementary proportional to the internal junction temperature of the chip. These analog voltage signals are converted into digital values by an on-chip ADC. The temperature value is obtained from a linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programmed into flash memory.

33.3 Temperature formula

Table 378. Calibration constants

Constant(1)	Description(2)
P1	Code from the ADC converting PTAT output voltage at 150^0C
P2	Code from the ADC converting PTAT output voltage at -40^0C

Table 378. Calibration constants (continued)

Constant(1)	Description(2)
C1	Code from the ADC converting CTAT output voltage at 150^0C
C2	Code from the ADC converting CTAT output voltage at -40^0C

1. Flash locations for P2, C2, P1 and C1 are located in UTEST sector, respectively at offset 0x00, 0x02, 0x04 and 0x06. Refer to DCF chapter for details.
2. It is mandatory to keep the ADC reference constant while measuring P1, P2, C1, and C2.

33.3.1 Equations for converting TSENS voltages to junction temperature

In the equations below:

- Pn and Cn are the calibration constants described in *Table 378*
- T is the unknown device temperature in ^{0}C
- Px is the code from the ADC converting PTAT output voltage at the temperature T with any ADC reference voltage Vref
- Cx is the code from the ADC converting CTAT output voltage at the temperature T with the same ADC reference voltage Vref.
- T2 = -40^0C
- T1 = 150^0C

Equation 16 $$A = Px \cdot C2 - (Cx \cdot P2)$$

Equation 17 $$B = Cx \cdot P1 - (Px \cdot C1)$$

The temperature T is calculated as

Equation 18 $$T = T2 + \frac{A \cdot (T1 - T2)}{A + B}$$

Caution: It is mandatory to have the same value as the ADC reference, while measuring Px and Cx.

33.3.2 Equations for converting TSENS voltages into constant reference (Digital bandgap voltage)

By converting the two outputs of the Temperature Sensor through ADC, one can find out a constant reference code (REFCODE) which does not change with temperature.

- Pn and Cn are the calibration constants described in *Table 378*.
- Px is the code from the ADC converting PTAT output voltage at any temperature T with any ADC reference voltage Vref.
- Cx is the code from the ADC converting CTAT output voltage at the same temperature T with the same ADC reference voltage Vref.

Equation 19

$$REFCODE = Px + \left(\frac{(P2 - P1)}{(C1 - C2)}\right) \times Cx$$

Caution: It is mandatory to measure Px and Cx at the same ADC reference voltage and at the same temperature.

Note: Software calculation must be made with negligible precision loss associated with limited bit representation of intermediate calculated results.

34 High-resolution timer (HRTIM)

34.1 Introduction

The high-resolution timer can generate up to 12 digital signals with highly accurate timings. It is primarily intended to drive power conversion systems such as switch mode power supplies or lighting systems, but can be of general purpose usage, whenever a very fine timing resolution is expected.

Its modular architecture allows to generate either independent or coupled waveforms. The wave-shape is defined by self-contained timings (using counters and compare units) and a broad range of external events, such as analog or digital feedbacks and synchronization signals. This allows to produce a large variety of control signals (PWM, phase-shifted, constant Ton,...) and address most of conversion topologies.

For control and monitoring purposes, the timer has also timing measure capabilities and links to built-in ADC and DAC converters. Last, it features light-load management mode and is able to handle various fault schemes for safe shut-down purposes.

34.2 Main features

- High-resolution timing units
  - 104 ps resolution, compensated against voltage and temperature variations
  - High-resolution available on all outputs, possibility to adjust duty-cycle, frequency and pulse width in triggered one-pulse mode
  - 6 16-bit timing units (each one with an independent counter and 4 compare units)
  - 12 outputs that can be controlled by any timing unit, up to 32 set/reset sources per channel
  - Modular architecture to address either multiple independent converters with 1 or 2 switches or few large multi-switch topologies
- Up to 10 external events, available for any timing unit
  - Programmable polarity and edge sensitivity
  - 5 events with a fast asynchronous mode
  - 5 events with a programmable digital filter
  - Spurious events filtering with blanking and windowing modes
- Multiple links to built-in analog peripherals
  - 4 triggers to ADC converters
  - 3 triggers to DAC converters
  - 3 comparators for analog signal conditioning
- Versatile protection scheme
  - 6 fault inputs can be combined and associated to any timing unit
  - Programmable polarity, edge sensitivity, and programmable digital filter
  - Dedicated delayed protections for resonant converters
- Multiple HRTIM instances can be synchronized with external synchronization inputs/outputs

- Versatile output stage
  - High-resolution deadtime insertion (down to 417 ps)
  - Programmable output polarity
  - Chopper mode
- Burst mode controller to handle light-load operation synchronously on multiple converters
- 8 interrupt vectors, each one with up to 14 sources
- 7 DMA requests with up to 14 sources, with a burst mode for multiple registers update

34.3 Functional description

34.3.1 General description

The HRTIM can be partitioned into several sub entities:

- The master timer
- The timing units (timer A to timer F)
- The output stage
- The burst mode controller
- An external event and fault signal conditioning logic that is shared by all timers
- The system interface

The master timer is based on a 16-bit up counter. It can set/reset any of the 12 outputs via 4 compare units and it provides synchronization signals to the 6 timer units. Its main purpose is to have the timer units controlled by a unique source. An interleaved buck converter is a typical application example where the master timer manages the phase-shifts between the multiple units.

The timer units are working either independently or coupled with the other timers including the master timer. Each timer contains the controls for two outputs. The output set/reset events are triggered either by the timing units compare registers or by events coming from the master timer, from the other timers or from external events.

The output stage has several duties

- Addition of deadtime when the 2 outputs are configured in complementary PWM mode
- Addition of a carrier frequency on top of the modulating signal
- Management of fault events, by asynchronously asserting the outputs to a predefined safe level

The burst mode controller can take over the control of one or multiple timers in case of light-load operation. The burst length and period can be programmed, as well as the idle state of the outputs.

The external event and fault signal conditioning logic includes:

- The input selection MUXes (for instance for selecting a digital input or an on-chip source for a given external event channel)
- Polarity and edge-sensitivity programming
- Digital filtering (for 5 channels out of 12)

The system interface allows the HRTIM to interact with the rest of the MCU:

- Interrupt requests to the CPU
- DMA controller for automatic accesses to/from the memories, including an HRTIM specific burst mode
- Triggers for the ADC and DAC converters

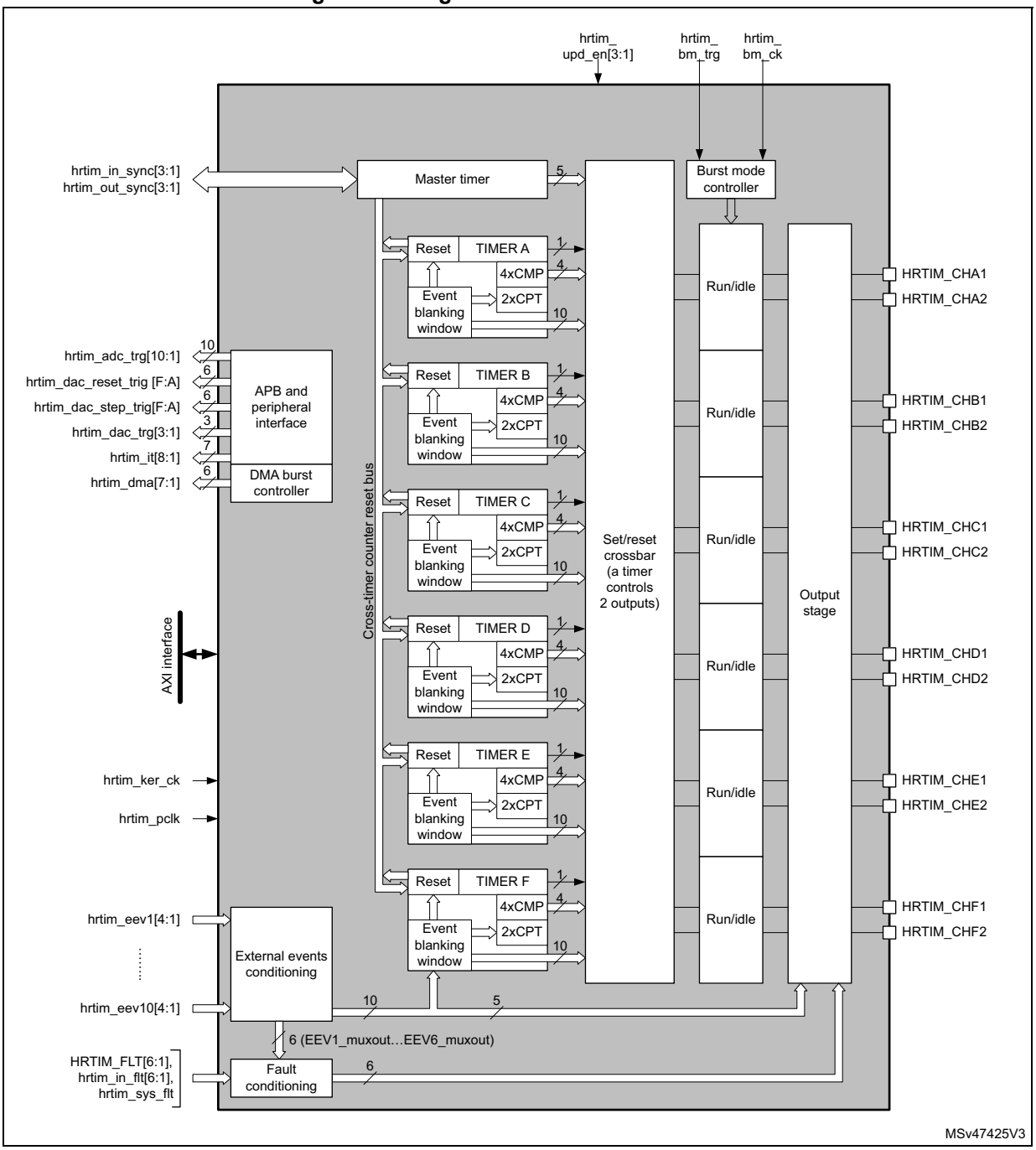
The HRTIM registers are split into 8 groups:

- Master timer registers
- Timer A to timer F registers
- Common registers for features shared by all timer units

Note: As a writing convention, references to the 6 timing units in the text and in registers are generalized using the "x" letter, where x can be any value from A to F.

The block diagram of the timer is shown in *Figure 337*.

Figure 337. High-resolution timer overview



34.3.2 HRTIM pins and internal signals

The tables in this section summarize the HRTIM inputs and outputs, both on-chip and off-chip.

Table 379. HRTIM inputs/outputs summary

Signal name	Signal type	Description
HRTIM_CHA1, HRTIM_CHA2, HRTIM_CHB1, HRTIM_CHB2, HRTIM_CHC1, HRTIM_CHC2, HRTIM_CHD1, HRTIM_CHD2, HRTIM_CHE1, HRTIM_CHE2, HRTIM_CHF1, HRTIM_CHF2	Outputs	Main HRTIM timer outputs. They can be coupled by pairs (HRTIM_CHx1 & HRTIM_CHx2) with deadtime insertion or work independently.
hrtim_in_flt1[4:1] hrtim_in_flt2[4:1] hrtim_in_flt3[4:1] hrtim_in_flt4[4:1] hrtim_in_flt5[4:1] hrtim_in_flt6[4:1]	Digital input	Fault inputs: immediately disable the HRTIM outputs when asserted (12 on-chip inputs and 6 off-chip HRTIM_FLTx inputs).
hrtim_sys_flt	Digital input	System fault gathering MCU internal fault events via Fault Collection and Control Unit.
hrtim_in_sync[3:1]	Digital input	Synchronization inputs to synchronize the whole HRTIM with other internal or external timer resources: hrtim_in_sync1: HRTimer 1: reserved HRTimer 2: hrtim1_scout1 hrtim_in_sync2: the source is the TIM1_TRGO output hrtim_in_sync3: the source is HRTIM_SCIN input pins
hrtim_out_sync[2:1]	Digital output	The purpose of this output is to cascade or synchronize several HRTIM instances, either on-chip or off-chip: hrtim_out_sync1: Refer to *Chapter 5: Device configuration*. hrtim_out_sync2: the destination is an off-chip HRTIM or peripheral (via HRTIM_SCOUT output pins, refer to *Chapter 5: Device configuration*).

Table 379. HRTIM inputs/outputs summary (continued)

Signal name	Signal type	Description
hrtim_eev1[4:1]	Digital input	External events. Each of the 10 events can be selected among 4 sources, either on-chip (from other built-in peripherals: comparator, ADC analog watchdog, TIMx timers, trigger outputs) or off-chip (HRTIM_EEVx input pins).
hrtim_eev2[4:1]		
hrtim_eev3[4:1]		
hrtim_eev4[4:1]		
hrtim_eev5[4:1]		
hrtim_eev6[4:1]		
hrtim_eev7[4:1]		
hrtim_eev8[4:1]		
hrtim_eev9[4:1]		
hrtim_eev10[4:1]		
hrtim_upd_en[3:1]	Digital input	A pulse on the update enable inputs hrtim_upd_end[3:1] (on-chip interconnect) triggers the transfer from shadow to active registers.
hrtim_bm_trg	Digital input	A pulse on this input triggers a burst mode entry. This input is connected to the TIM7_TRGO output.
hrtim_bm_ck[4:1]	Digital input	Burst mode clock (on-chip interconnect)
hrtim_adc_trg[10:1]	Digital output	ADC start of conversion triggers. The hrtim_adc_trg1.. hrtim_adc_trg10 outputs are here-after referred to as ADC trigger 1 to ADC trigger 10.
hrtim_dac_trg[3:1]	Digital output	DAC conversion update triggers
hrtim_dac_reset_trg[F:A] hrtim_dac_step_trg[F:A]	Digital output	Dual channel DAC triggers
hrtim_it[8:1]	Digital output	Interrupt requests
hrtim_dma[7:1]	Digital output	DMA requests
hrtim_pclk	-	HRTIM bus interface clock (SYS_CLK clock)
hrtim_ker_ck	-	HRTIM kernel clock (SYS_CLK clock)

For external events mapping and associated features, refer to the HRTIM external events mapping and associated features tables of *Chapter 5: Device configuration*.

Table 380. Update enable inputs and sources

hrtim_upd_en[3:1]	Update source
hrtim_upd_en1	TIM16_OC
hrtim_upd_en2	User controlled signal, refer to *Chapter 21: System configuration controller (SYSCFG)*
hrtim_upd_en3	TIM6_TRGO

Table 381. Burst mode clock sources from general purpose timer

hrtim_bm_ck[4:1]	BMCLK[3:0]	Clock source
hrtim_bm_ck1	0110	TIM16 OC
hrtim_bm_ck2	0111	Reserved
hrtim_bm_ck3	1000	TIM7 TRGO
hrtim_bm_ck4	1001	Reserved

For faults inputs per HRTIM instance, refer to the HRTIM fault inputs tables of *Chapter 5: Device configuration*.

For HRTIM DAC triggers connections, refer to the HRTIM DAC triggers connections tables of *Chapter 5: Device configuration*.

34.3.3 Clocks

The HRTIM must be supplied by the t_{HRTIM} SYSTEM_CLK clock to offer a full resolution. The t_{HRTIM} clock period is evenly divided into up to 32 intermediate steps using an edge positioning logic. All clocks present in the HRTIM are derived from this reference clock.

Definition of terms

f_{HRTIM}: main HRTIM clock (hrtim_ker_ck). All subsequent clocks are derived and synchronous with this source.

f_{HRCK}: high-resolution equivalent clock. Considering the f_{HRTIM} clock period division by 32, it is equivalent to a frequency of 300 x 32 = 9.6 GHz.

f_{DTG}: deadtime generator clock. For convenience, only the t_{DTG} period ($t_{DTG} = 1/f_{DTG}$) is used in this document.

f_{CHPFRQ}: chopper stage clock source.

f_{1STPW}: clock source defining the length of the initial pulse in chopper mode. For convenience, only the t_{1STPW} period ($t_{1STPW} = 1/f_{1STPW}$) is used in this document.

f_{BRST}: burst mode controller counter clock.

$f_{SAMPLING}$: clock needed to sample the fault or the external events inputs.

f_{FLTS}: clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter fault events.

f_{EEVS}: clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter external events.

Timer clock and prescaler

Each timer in the HRTIM has its own individual clock prescaler, which allows you to adjust the timer resolution (refer to *Table 382*).

Table 382. Timer resolution and min. PWM frequency for f_{HRTIM} = 300 MHz

CKPSC[2:0]	Prescaling ratio	f_{HRCK} Equivalent frequency	Resolution	Min PWM frequency
000	1	300 x 32 MHz = 9.6 GHz	104 ps	146.6 kHz
001	2	300 x 16 MHz = 4.8 GHz	208 ps	73.26 kHz
010	4	300 x 8 MHz = 2.4 GHz	416 ps	36.63 kHz
011	8	300 x 4 MHz = 1.2 GHz	833 ps	18.31 kHz
100	16	300 x 2 MHz = 600 MHz	1.66 ns	9.16 kHz
101	32	300 MHz	3.33 ns	4.58 kHz
110	64	300/2 MHz = 150 MHz	6.66 ns	2.29 kHz
111	128	300/4 MHz = 75 MHz	13.33 ns	1.14 kHz

The high-resolution is available for edge positioning, PWM period adjustment and externally triggered pulse duration.

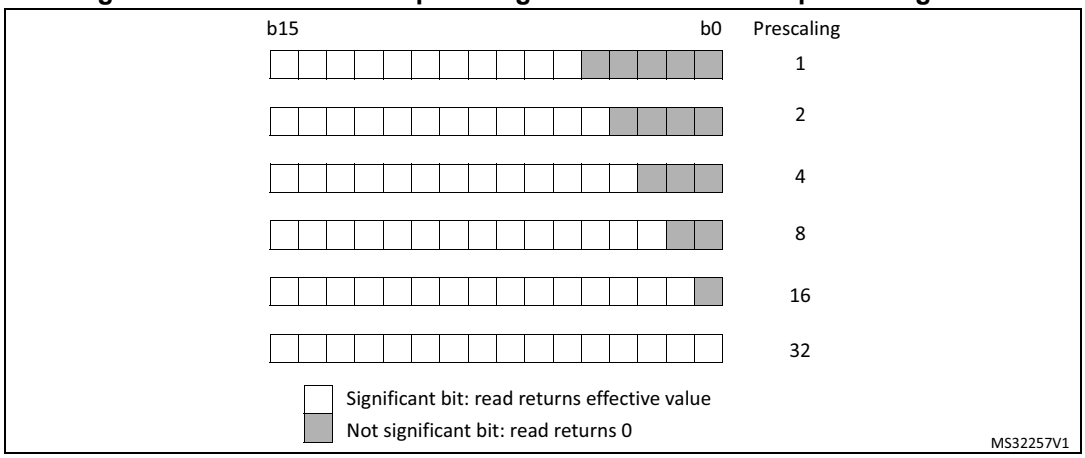
The high-resolution is not available for the following features

- Timer counter read and write accesses
- Capture unit

For clock prescaling ratios below 32 (CKPSC[2:0] < 5), the least significant bits of the counter and capture registers are not significant. The least significant bits cannot be written (counter register only) and return 0 when read.

For instance, if CKPSC[2:0] = 2 (prescaling by 4), writing 0xFFFF into the counter register yields an effective value of 0xFFF8. Conversely, any counter value between 0xFFFF and 0xFFF8 is read as 0xFFF8.

Figure 338. Counter and capture register format vs clock prescaling factor



Initialization

At start-up, it is mandatory to initialize first the prescaler bit fields before writing the compare and period registers. Once the timer is enabled (MCEN or TxCEN bit set in the HRTIM_MCR register), the prescaler cannot be modified.

When multiple timers are enabled, the prescalers are synchronized with the prescaler of the timer that was started first.

Warning: It is possible to have different prescaling ratios in the master and TIMA..E timers only if the counter and output behavior does not depend on other timers' information and signals. It is mandatory to configure identical prescaling ratios in these timers when one of the following events is propagated from one timing unit (or master timer) to another: output set/reset event, counter reset event, update event, external event filter or capture triggers. Prescaler factors not equal yield to unpredictable results.

Deadtime generator clock

The deadtime prescaler is supplied by (f_{HRTIM} x 8) / $2^{(DTPRSC[2:0])}$, programmed with DTPRSC[2:0] bits in the HRTIM_DTxR register.

t_{DTG} ranges from 417 ps to 53.33 ns for f_{HRTIM} = 300 MHz.

Chopper stage clock

The chopper stage clock source f_{CHPFRQ} is derived from f_{HRTIM} with a division factor ranging from 16 to 256, so that 1.172 MHz ≤ f_{CHPFRQ} ≤ 18.75 MHz for f_{HRTIM} = 300 MHz.

t_{1STPW} is the length of the initial pulse in chopper mode, programmed with the STRPW[3:0] bits in the HRTIM_CHPxR register, as follows:

t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.

It uses f_{HRTIM} / 16 as clock source (18.75 MHz for f_{HRTIM}= 300 MHz).

Burst mode prescaler

The burst mode controller counter clock f_{BRST} can be supplied by several sources, among which one is derived from f_{HRTIM}.

In this case, f_{BRST} ranges from f_{HRTIM} to f_{HRTIM} / 32768 (9.155 kHz for f_{HRTIM} = 300 MHz).

Fault input sampling clock

The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{FLTS}.

f_{FLTS} is derived from f_{HRTIM} and ranges from $f_{HRTIM\ to}$ f_{HRTIM} / 8 (300 MHz to 37.5 MHz for f_{HRTIM} = 300 MHz).

External event input sampling clock

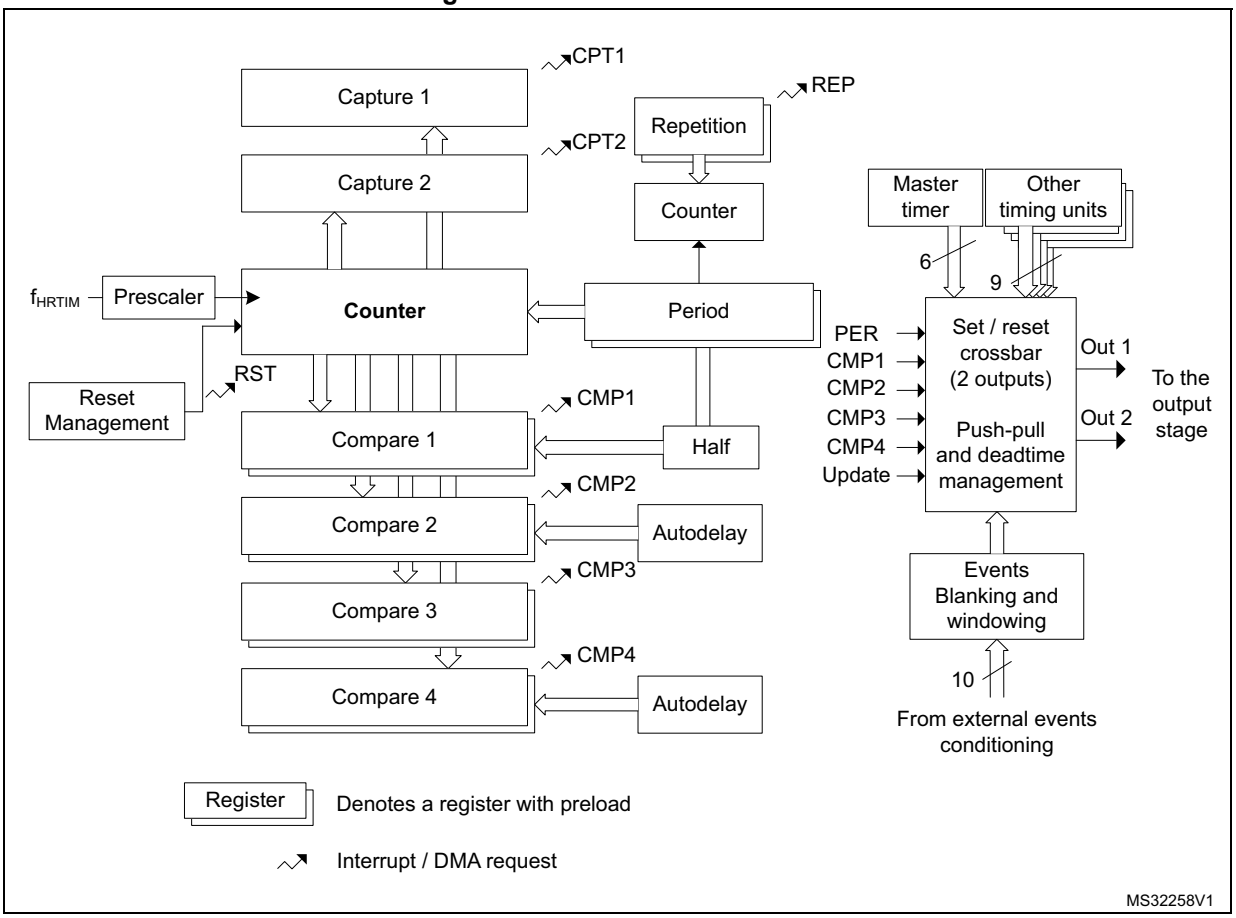
The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{EEVS}.

f_{EEVS} is derived from f_{HRTIM} and ranges from $f_{HRTIM\ to}$ f_{HRTIM} / 8 (300 MHz to 37.5 MHz for f_{HRTIM} = 300 MHz.

34.3.4 Timer A..F timing units

The HRTIM embeds 6 identical timing units made of a 16-bit up-counter with an auto-reload mechanism to define the counting period, 4 compare and 2 capture units, as per *Figure 339*. Each unit includes all control features for 2 outputs, so that it operates as a standalone timer.

Figure 339. Timer A..F overview



The period and compare values must be within a lower and an upper limit related to the high-resolution implementation and listed in *Table 383*:

- The minimum value must be greater than or equal to 3 periods of the f_{HRTIM} clock. The value 0x0000 can be written in CMP1 and CMP3 registers only, to skip a PWM pulse. Refer to *Null duty cycle exception case* for details.
- The maximum value must be less than or equal to 0xFFFF - 1 periods of the f_{HRTIM} clock.

Table 383. Period and compare registers min and max values

CKPSC[2:0] value	Min(1)	Max
0	0x0060	0xFFDF
1	0x0030	0xFFEF
2	0x0018	0xFFF7
3	0x000C	0xFFFB
4	0x0006	0xFFFD
≥ 5	0x0003	0xFFFD

1. The value 0x0000 can be written in CMP1 and CMP3 registers only, to skip a PWM pulse. Refer to *Null duty cycle exception case* for details.

Note: A compare value greater than the period register value does not generate a compare match event.

Counter operating mode

Timer A..F operates in continuous (free-running) mode or in single-shot manner where counting is started by a reset event, using the CONT bit in the HRTIM_TIMxCR control register. An additional RETRIG bit allows you to select whether the single-shot operation is retriggerable or non-retriggerable. Details of operation are summarized in *Table 384* and in *Figure 340* and *Figure 341*.

Table 384. Timer operating modes

CONT	RETRIG	Operating mode	Start / stop conditions Clocking and event generation
0	0	Single-shot Non-retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A first reset event starts the counting and any subsequent reset is ignored until the counter reaches the PER value. The PER event is then generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
0	1	Single-shot Retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A reset event starts the counting if the counter is stopped, otherwise it clears the counter. When the counter reaches the PER value, the PER event is generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
1	X	Continuous mode	Setting the TxEN bit enables the timer and starts the counter simultaneously. When the counter reaches the PER value, it rolls-over to 0x0000 and resumes counting. The counter can be reset at any time.

The TxEN bit can be cleared at any time to disable the timer and stop the counting.

Figure 340. Continuous timer operation

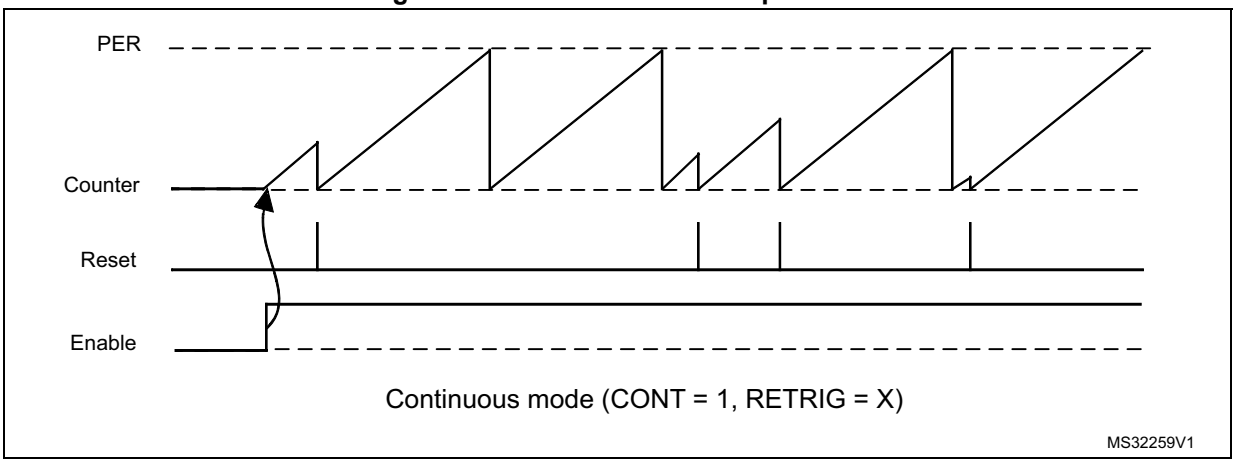
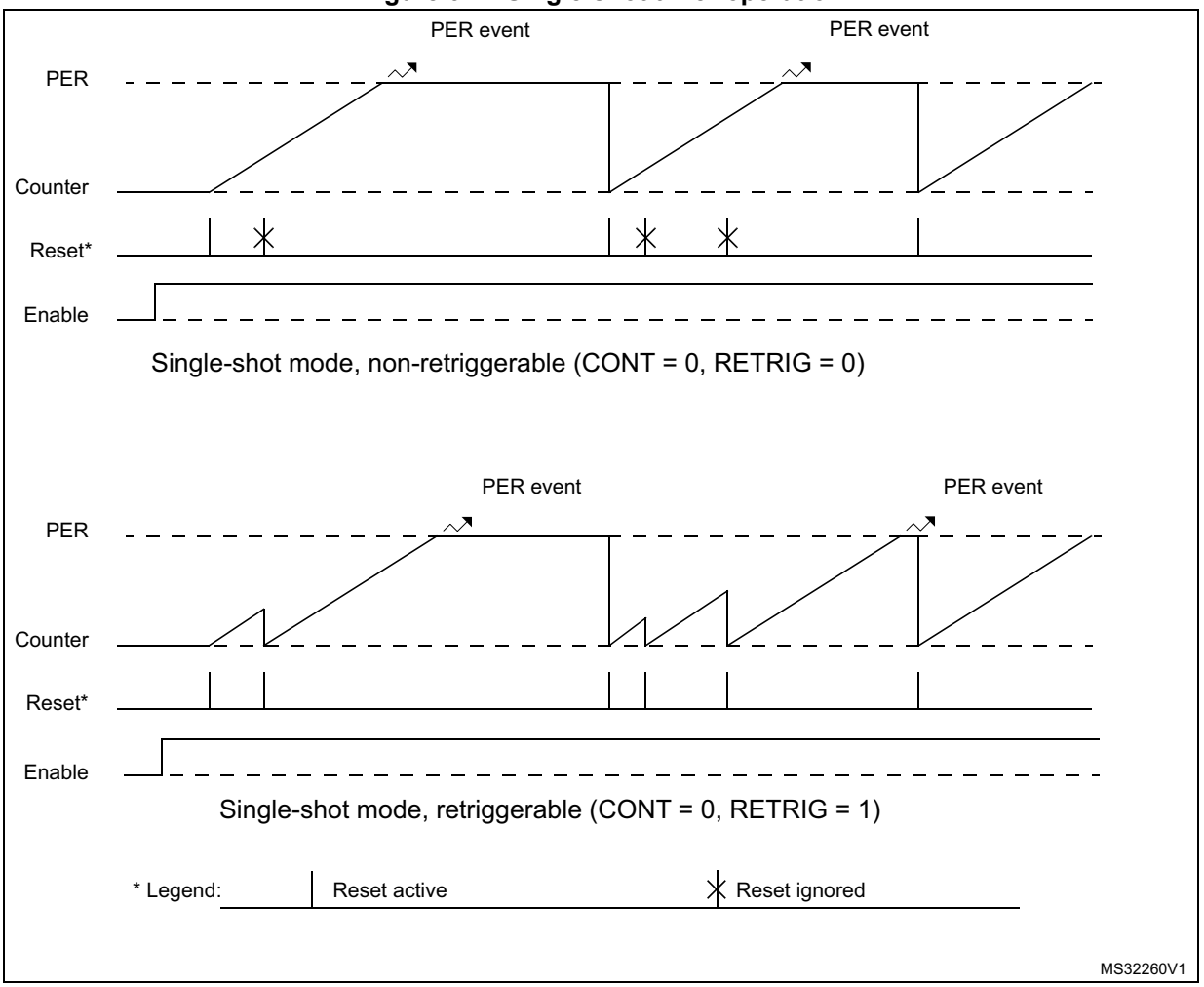


Figure 341. Single-shot timer operation



Roll-over event

A counter roll-over event is generated when the counter goes back to 0 after having reached the period value set in the HRTIM_PERxR register in continuous mode.

This event is used for multiple purposes in the HRTIM:

- To set/reset the outputs
- To trigger the register content update (transfer from preload to active)
- To trigger an IRQ or a DMA request
- To serve as a burst mode clock source or a burst start trigger
- As an ADC trigger
- To decrement the repetition counter

If the initial counter value is above the period value when the timer is started, or if a new period is set while the counter is already above this value, the counter is not reset: it overflows at the maximum period value and the repetition counter does not decrement.

Timer reset

The reset of the timing unit counter can be triggered by up to 32 events that can be selected simultaneously in the HRTIM_RSTxR register, among the following sources:

- The timing unit: compare 2, compare 4 and update (3 events)
- The master timer: reset and compare 1..4 (5 events)
- The external events EXTEVNT1..10 (10 events)
- All other timing units (for example, timer B..F for timer A): compare 1, 2 and 4 (14 events)

Several events can be selected simultaneously to handle multiple reset sources. In this case, the multiple reset requests are ORed. When 2 counter reset events are generated within the same f_{HRTIM} clock cycle, the last counter reset is taken into account.

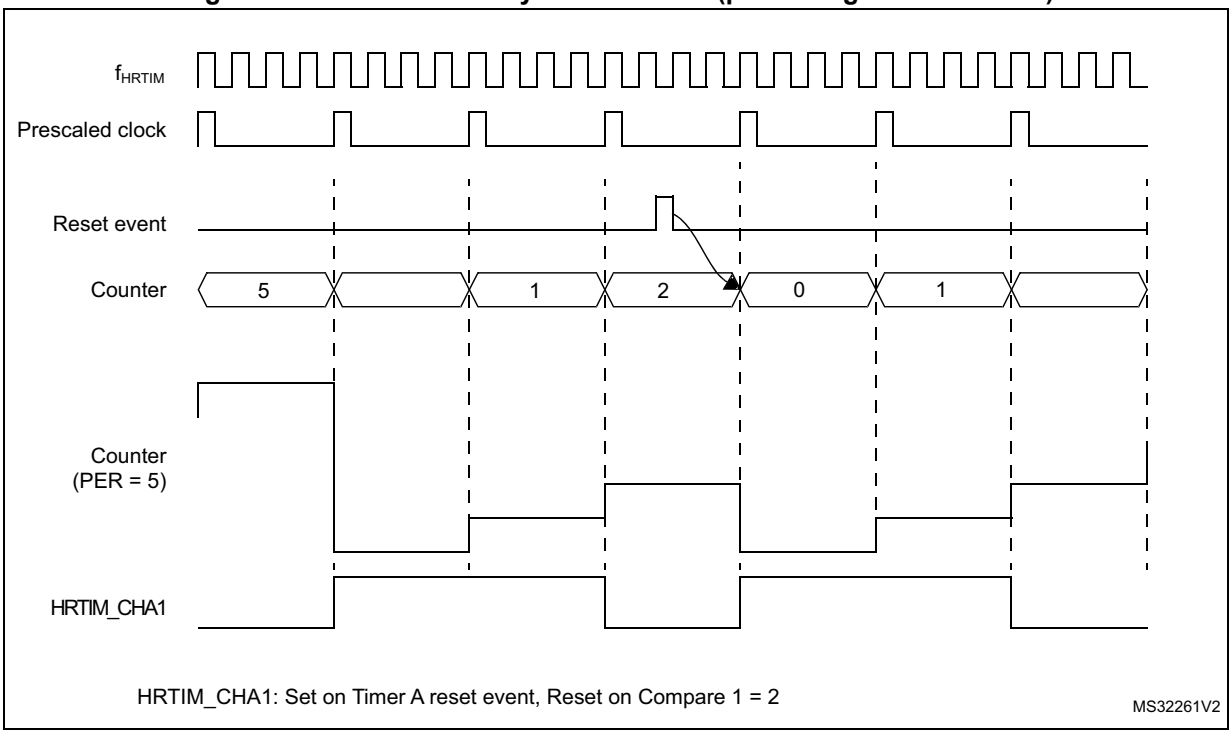
Additionally, it is possible to do a software reset of the counter using the TxRST bits in the HRTIM_CR2 register. These control bits are grouped into a single register to allow the simultaneous reset of several counters.

The reset requests are taken into account only once the related counters are enabled (TxCEN bit set).

When the f_{HRTIM} clock prescaling ratio is above 32 (counting period above f_{HRTIM}), the counter reset event is delayed to the next active edge of the prescaled clock. This allows to maintain a jitterless waveform generation when an output transition is synchronized to the reset event (typically a constant Ton time converter).

Figure 342 shows how the reset is handled for a clock prescaling ratio of 128 (f_{HRTIM} divided by 4).

Figure 342. Timer reset resynchronization (prescaling ratio above 32)



Repetition counter

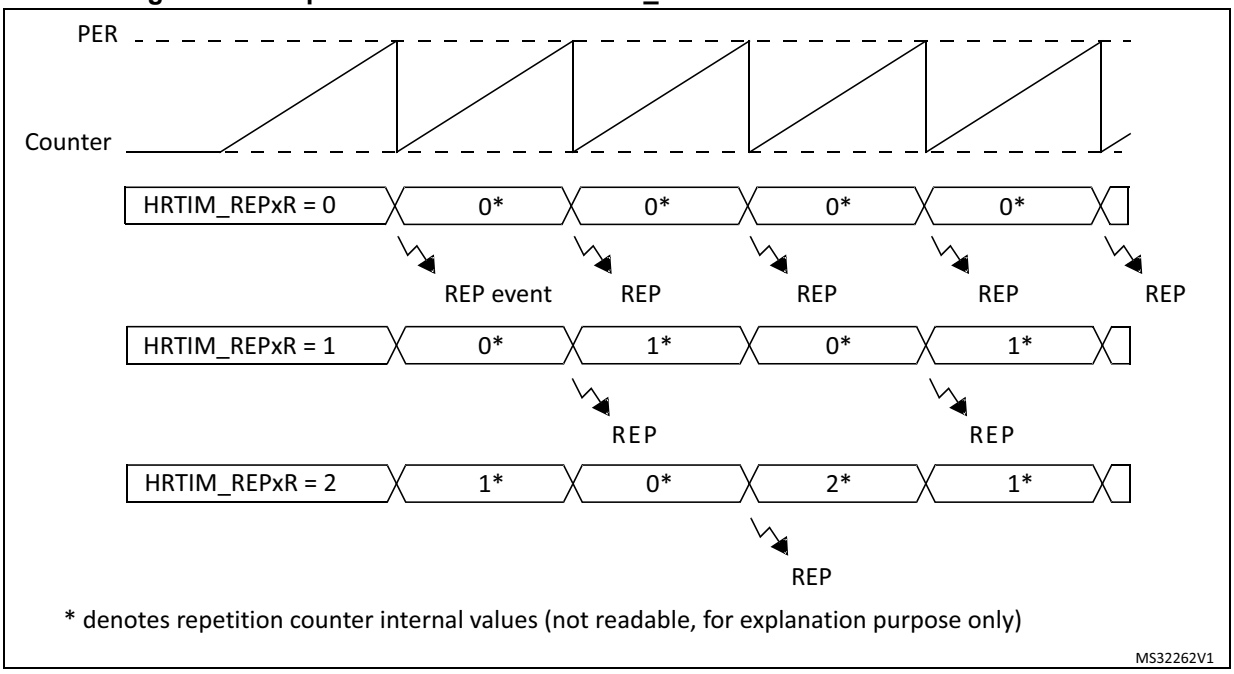
A common software practice is to have an interrupt generated when the period value is reached, so that the maximum amount of time is left for processing before the next period begins. The main purpose of the repetition counter is to adjust the period interrupt rate and off-load the CPU by decoupling the switching frequency and the interrupt frequency.

The timing units have a repetition counter. This counter cannot be read, but solely programmed with an auto-reload value in the HRTIM_REPxR register.

The repetition counter is initialized with the content of the HRTIM_REPxR register when the timer is enabled (TXCEN bit set). Once the timer has been enabled, any time the counter is cleared, either due to a reset event or due to a counter roll-over, the repetition counter is decreased. When it reaches zero, a REP interrupt or a DMA request is issued if enabled (REPIE and REPDE bits in the HRTIM_DIER register).

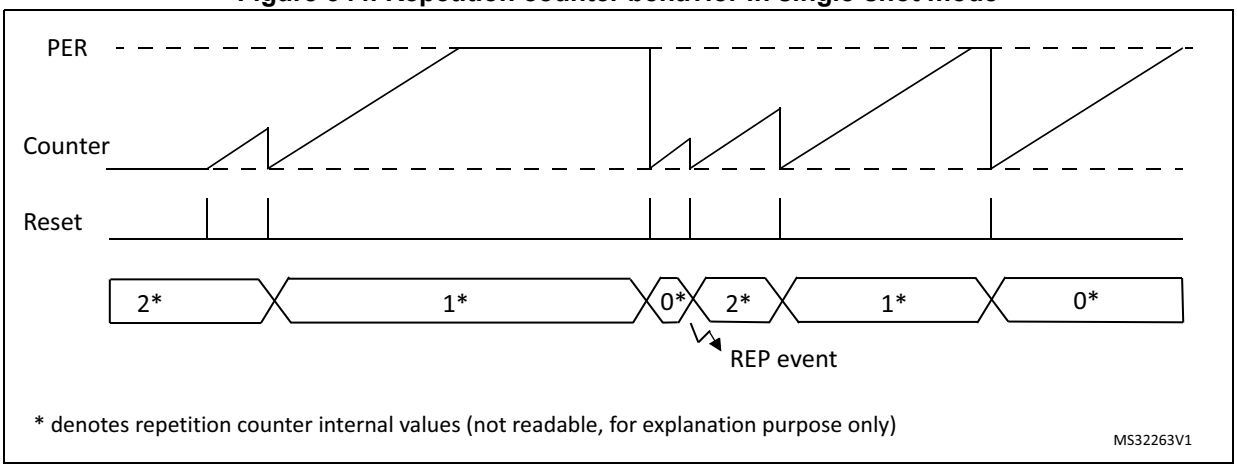
If the HRTIM_REPxR register is set to 0, an interrupt is generated for each and every period. For any value above 0, a REP interrupt is generated after (HRTIM_REPxR + 1) periods. *Figure 343* presents the repetition counter operation for various values, in continuous mode.

Figure 343. Repetition rate versus HRTIM_REPxR content in continuous mode



The repetition counter can also be used when the counter is reset before reaching the period value (variable frequency operation) either in continuous or in single-shot mode (refer to *Figure 344*). The reset causes the repetition counter to be decremented, at the exception of the very first start following counter enable (TxCEN bit set).

Figure 344. Repetition counter behavior in single-shot mode



A reset or start event from the HRTIM_SCIN[3:1] source causes the repetition to be decremented as any other reset. However, in SYNCIN-started single-shot mode (SYNCSTRTx bit set in the HRTIM_TIMxCR register), the repetition counter is decremented only on the 1st reset event following the period. Any subsequent reset does not alter the repetition counter until the counter is re-started by a new request on HRTIM_SCIN[3:1] inputs.

Set/reset crossbar

A “set” event corresponds to a transition to the output active state, while a “reset” event corresponds to a transition to the output inactive state.

The polarity of the waveform is defined in the output stage to accommodate positive or negative logic external components: an active level corresponds to a logic level 1 for a positive polarity (POLx = 0), and to a logic level 0 for a negative polarity (POLx = 1).

Each of the timing units handles the set/reset crossbar for two outputs. These 2 outputs can be set, reset or toggled by up to 32 events that can be selected among the following sources:

- The timing unit: period, compare 1..4, register update (6 events)
- The master timer: period, compare 1..4, HRTIM synchronization (6 events)
- All other timing units (for example, timer B..F for timer A): TIMEVNT1..9 (9 events described in *Table 385*)
- The external events EXTEVNT1..10 (10 events)
- A software forcing (1 event)

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

The event sources are ORed and multiple events can be simultaneously selected.

Each output is controlled by two 32-bit registers, one coding for the set (HRTIM_SETxyR) and another one for the reset (HRTIM_RSTxyR), where x stands for the timing unit: A..F and y stands for the output 1or 2 (for example, HRTIM_SETA1R, HRTIM_RSTC2R,...).

If the same event is selected for both set and reset, it toggles the output. It is not possible to toggle the output state more than one time per t_{HRTIM} period: in case of two consecutive toggling events within the same cycle, only the first one is considered.

The set and reset requests are taken into account only once the counter is enabled (TxCEN bit set), except if the software is forcing a request to allow the prepositioning of the outputs at timer start-up.

Table 385 summarizes the events from other timing units that can be used to set and reset the outputs. The number corresponds to the timer events (such as TIMEVNTx) listed in the register, and empty locations are indicating non-available events.

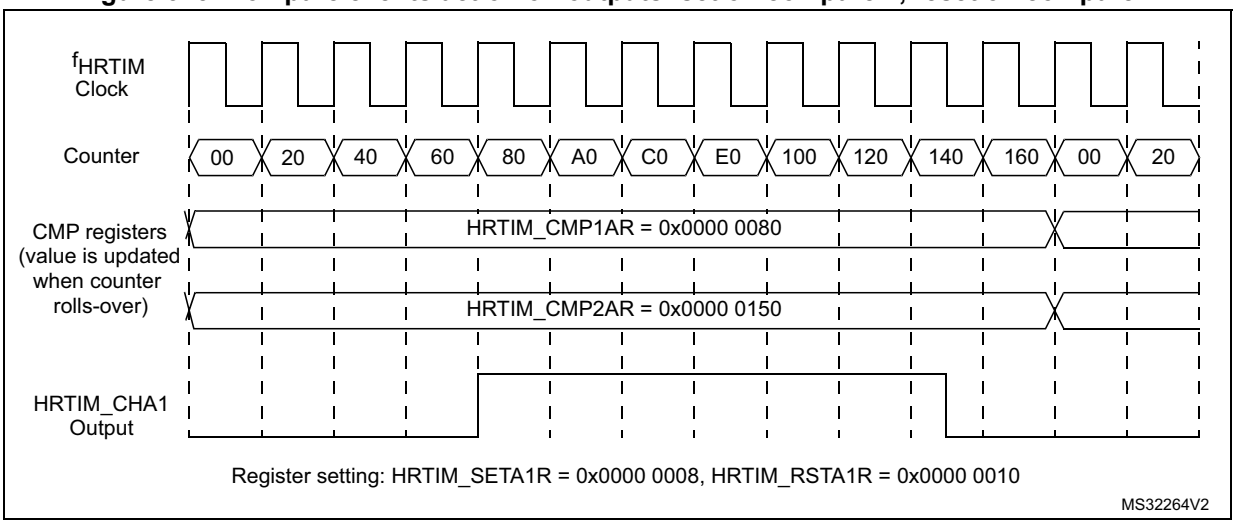
For instance, timer A outputs can be set or reset by the following events: timer B compare 1 and 2, timer C compare 2 and 3,... and timer E compare 3 is listed as TIMEVNT7 in HRTIM_SETA1R.

Table 385. Events mapping across timer A to F

Source		Timer A				Timer B				Timer C				Timer D				Timer E				Timer F			
		CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4
Destination	TA	-	-	-	-	1	2	-	-	-	3	4	-	5	6	-	-	-	-	7	8	-	-	-	9
	TB	1	2	-	-	-	-	-	-	-	-	3	4	-	-	5	6	7	8	-	-	-	-	9	-
	TC	-	1	2	-	-	3	4	-	-	-	-	-	-	5	-	6	-	-	7	8	-	9	-	-
	TD	1	-	-	2	-	3	-	4	-	-	-	5	-	-	-	-	6	-	-	7	8	-	9	-
	TE	-	-	-	1	-	-	2	3	4	5	-	-	6	7	-	-	-	-	-	-	-	-	8	9
	TF	-	-	1	-	2	-	-	3	4	-	-	5	-	-	6	7	-	8	9	-	-	-	-	-

Figure 345 represents how a PWM signal is generated using two compare events.

Figure 345. Compare events action on outputs: set on compare 1, reset on compare 2



Set/reset on update events

A set or reset event on update is done at low resolution. When CKPSC[2:0] < 5, the high-resolution delay is set to its maximum value so that a set/reset event on update always lags as compared to other compare set/reset events, with a jitter varying between 0 and 31/32 of a f_{HRTIM} clock period.

Half mode

This mode aims at generating square signal with fixed 50% duty cycle and variable frequency (typically for converters using resonant topologies). It allows to have the duty cycle automatically forced to half of the period value when a new period is programmed.

This mode is enabled by writing HALF bit to 1 in the HRTIM_TIMxCR register. When the HRTIM_PERxR register is written, it causes an automatic update of the compare 1 value with HRTIM_PERxR/2 value.

The output on which a square wave is generated must be programmed to have one transition on CMP1 event, and one transition on the period event, as follows:

– HRTIM_SETxyR = 0x0000 0008, HRTIM_RSTxyR = 0x0000 0004, or
– HRTIM_SETxyR = 0x0000 0004, HRTIM_RSTxyR = 0x0000 0008

The HALF mode overrides the content of the HRTIM_CMP1xR register. The access to the HRTIM_PERxR register only causes compare 1 internal register to be updated. The user-accessible HRTIM_CMP1xR register is not updated with the HRTIM_PERxR/2 value.

When the preload is enabled (PREEN = 1, MUDIS, TxUDIS), compare 1 active register is refreshed on the update event. If the preload is disabled (PREEN= 0), compare 1 active register is updated as soon as HRTIM_PERxR is written.

The period must be greater than or equal to 6 periods of the f_{HRTIM} clock (0xC0 if CKPSC[2:0] = 0, 0x60 if CKPSC[2:0] = 1, 0x30 if CKPSC[2:0] = 2,...) when the HALF mode is enabled.

Interleaved mode

This mode complements the Half mode and helps the implementation of interleaved topologies.

It allows to re-compute automatically the content of compare registers when the HRTIM_PERxR value is updated.

The selection is done using the HALF bit and the IL[1:0] bits in HRTIM_MCR and HRTIM_TIMxCR registers, as shown in *Table 386*.

Table 386. Interleaved mode selection

HALF bit	INTLVD [1:0] bits	Mode
0	00	Disabled
0	01	Triple interleaved (120°)
0	10	Quad interleaved (90°)
0	11	Reserved
1	xx	Dual interleaved (180°)

Table 387 gives the compare values for the 3 available modes. The content of the compare registers is overridden. The corresponding compare events can be used to trigger an output set/reset or to reset a slave timer.

Table 387. Compare 1..3 values in interleaved mode

Mode	Dual interleaved 180°	Triple interleaved 120°	Quad interleaved 90°
CMP1 value	PERxR/2	PERxR/3	PERxR/4
CMP2 value	Not affected	2x (PERxR/3)	PERxR/2
CMP3 value	Not affected	Not affected	3x (PERxR/4)

Note: In half and interleaved modes, the compare registers are controlled by hardware and writing them has no effect. However the written value is stored in the preload register and becomes active on the update event following the exit of these modes.

Note: The triple and quad interleaved modes must not be used simultaneously with other modes using CMP2 (dual channel dac trigger and triggered-half modes).

Null duty cycle exception case

The high-resolution behavior is not supported for pulses narrower than 3 t_{HRTIM} periods (refer to *Section 34.3.7: Set/reset event priorities and narrow pulse management*) and any value strictly below 3 periods of the f_{HRTIM} clock (that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...) in the HRTIM_TIMxCMPy register is forbidden (refer to *34.5.20: HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F)*).

However, it is possible to skip an output pulse and have a null duty cycle by simply writing a null value in the following two registers: HRTIM_TIMxCMP1 and HRTIM_TIMxCMP3, if and only if the following conditions are met:

- the output SET event is generated by the PERIOD event
- the output RESET if generated by the compare 1 (respectively compare 3) event
- the compare 1 (compare 3) event is active within the timer unit itself, and not used for other timing units

For any other use case, this can be done by programming the SET and RESET events with the very same compare values, above 3 periods of the f_{HRTIM} clock. In this case, the output is reset forced (following the set/reset priority scheme defined in the *Section 34.3.7: Set/reset event priorities and narrow pulse management*).

Swap mode

This mode allows to swap the two outputs with a single bit access: the output 1 signal is connected to the output 2 pin and the output 2 signal is connected to output 1 pin. The output swap is triggered with the SWPx bits in the HRTIM_CR2 register and is effective on the next update event.

The outputs are swapped prior to the set/reset crossbar unit, as follows:

- if SWPx = 0, HRTIM_SETx1R and HRTIM_RSTx1R are coding for the output 1, HRTIM_SETx2R and HRTIM_RSTx2R are coding for the output 2
- if SWPx = 1, HRTIM_SETx1R and HRTIM_RSTx1R are coding for the output 2, HRTIM_SETx2R and HRTIM_RSTx2R are coding for the output 1

The swap mode is only affecting the preload register, and not the active registers.

Note: The preload mode must be enabled when using the swap mode.

Consequently, it does not modify the auxiliary outputs in parallel with the regular outputs going to the output stage (refer to *Section 34.3.18* for details). They provide the following internal status, events and signals:

- O1CPY, O2CPY, SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests
- Capture triggers upon output set/reset (TA2, TB2, TC2, TD2, TE2, TF2)
- External event filters generated with a Tx2 output copy

For instance the SETx1 flag is related to the output 1 when SWP = 0 and is related to the output 2 when SWPx = 1.

Similarly, the swap mode does not change the attribution of control bits in the HRTIM_OUTxR register (DIDLx, CHPx, FAULTx[1:0], IDLESx, POLx bits). For instance, the POL1 bit controls the output 1 polarity whatever the SWP bit value.

Note: The SWPx bits are ignored in push-pull mode (PSHPLL = 1 in the HRTIM_TIMxCR register).

Capture

The timing unit has the capability to capture the counter value, triggered by internal and external events. The purpose is to:

- measure events arrival timings or occurrence intervals
- update compare 2 and compare 4 values in auto-delayed mode (refer to *Auto-delayed mode*).

The capture is done with f_{HRTIM} resolution: for a clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the register are not significant (read as 0).

The timer has 2 capture registers: HRTIM_CPT1xR and HRTIM_CPT2xR. The capture triggers are programmed in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers.

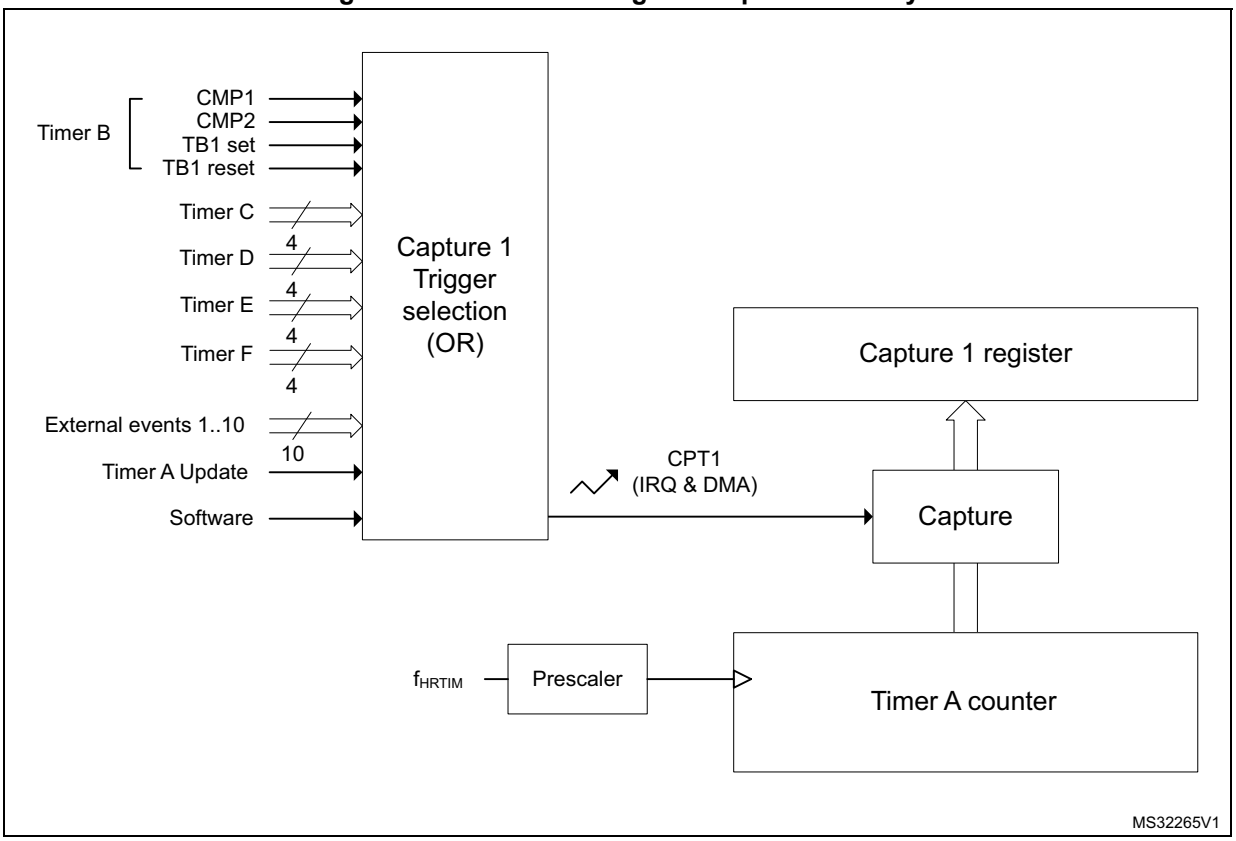
The capture of the timing unit counter can be triggered by up to 28 events that can be selected simultaneously in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers, among the following sources:

- The external events, EXTEVNT1..10 (10 events)
- All other timing units (for example, timer B..F for timer A): compare 1, 2 and output 1 set/reset events (16 events)
- The timing unit: update (1 event)
- A software capture (1 event)

Several events can be selected simultaneously to handle multiple capture triggers. In this case, the concurrent trigger requests are ORed. The capture can generate an interrupt or a DMA request when CPTxIE and CPTxDE bits are set in the HRTIM_TIMxDIER register.

Over-capture is not prevented by the circuitry: a new capture is triggered even if the previous value was not read, or if the capture flag was not cleared.

Figure 346. Timer A timing unit capture circuitry



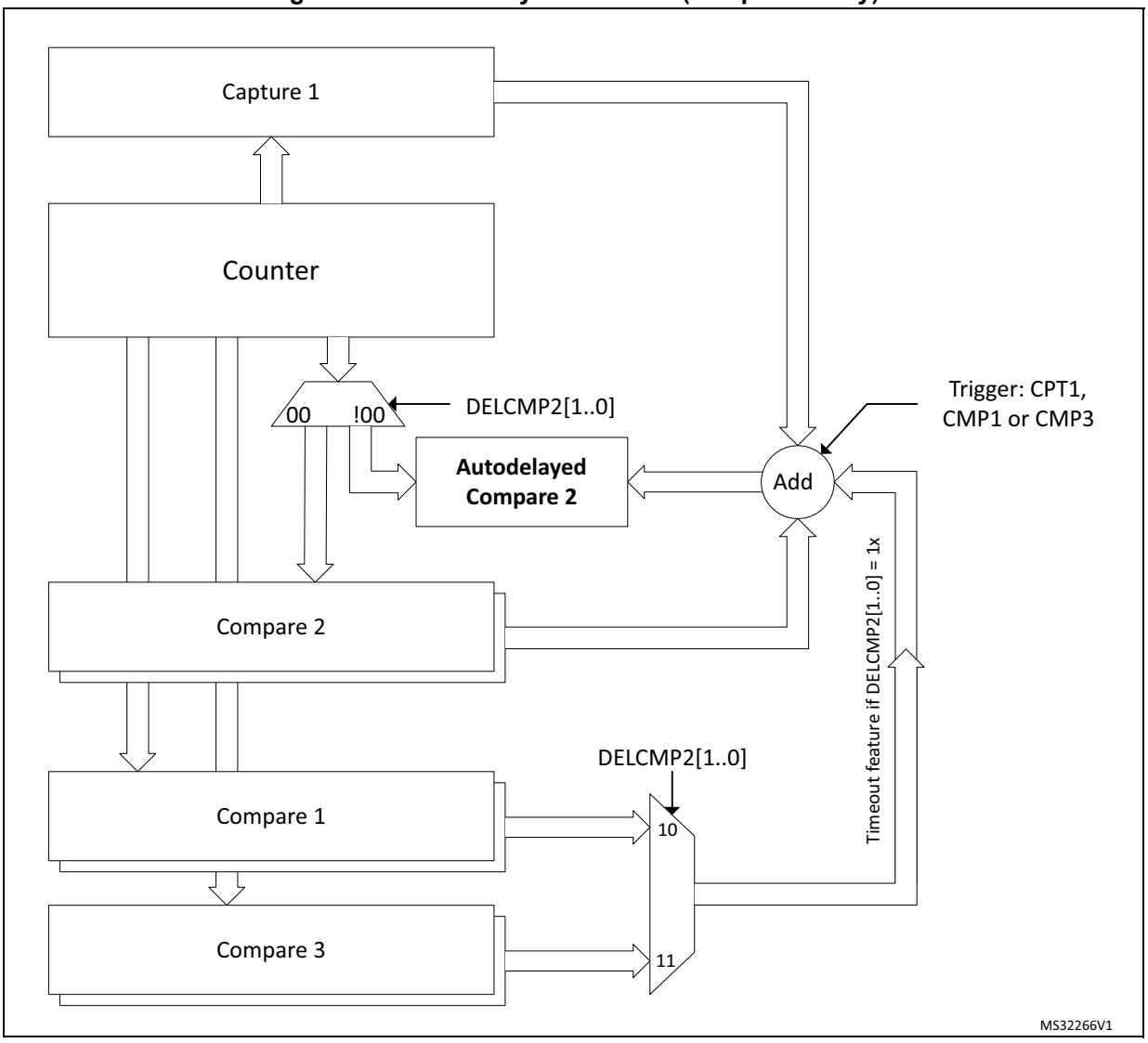
Auto-delayed mode

This mode allows to have compare events generated relatively to capture events, so that for instance an output change can happen with a programmed timing following a capture. In this case, the compare match occurs independently from the timer counter value. It enables the generation of waveforms with timings synchronized to external events without the need of software computation and interrupt servicing.

As long as no capture is triggered, the content of the HRTIM_CMPxR register is ignored (no compare event is generated when the counter value matches the compare value). Once the capture is triggered, the compare value programmed in HRTIM_CMPxR is summed with the captured counter value in HRTIM_CPTxyR, and it updates the internal auto-delayed compare register, as seen in *Figure 347*. The auto-delayed compare register is internal to the timing unit and cannot be read. The HRTIM_CMPxR preload register is not modified after the calculation.

This feature is available only for compare 2 and compare 4 registers. compare 2 is associated with capture 1, while compare 4 is associated with capture 2. HRTIM_CMP2xR and HRTIM_CMP4xR compares cannot be programmed with a value below 3 f_{HRTIM} clock periods, as in the regular mode.

Figure 347. Auto-delayed overview (compare 2 only)



The auto-delayed compare is only valid from the capture up to the period event: once the counter has reached the period value, the system is re-armed with compare disabled until a capture occurs.

DELCMP2[1:0] and DELCMP4[1:0] bits in HRTIM_TIMxCR register allow to configure the auto-delayed mode as follows:

- 00
  Regular compare mode: HRTIM_CMP2xR and HRTIM_CMP4xR register contents are directly compared with the counter value.
- 01
  Auto-delayed mode: compare 2 and compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event.

- 10 or 11
  Auto-delayed mode with timeout: compare 2 and compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event or after a compare 1 match (DELCMPx[1:0]= 10) or a compare 3 match (DELCMPx[1:0]= 11) to have a timeout function if capture 1/2 event is missing.

When the capture occurs, the comparison is done with the (HRTIM_CMP2/4xR + HRTIM_CPT1/2xR) value. If no capture is triggered within the period, the behavior depends on the DELCMPx[1:0] value:

- DELCMPx[1:0] = 01: the compare event is not generated
- DELCMPx[1:0] = 10 or 11: the comparison is done with the sum of the 2 compares (for instance HRTIM_CMP2xR + HRTIM_CMP1xR). The captures are not taken into account if they are triggered after CMPx + CMP1 (resp. CMPx + CMP3).

The captures are enabled again at the beginning of the next PWM period.

If the result of the auto-delayed summation is above 0xFFFF (overflow), the value is ignored and no compare event is generated until a new period is started.

Note: DELCMPx[1:0] bit field must be reset when reprogrammed from one value to the other to re-initialize properly the auto-delayed mechanism, for instance:

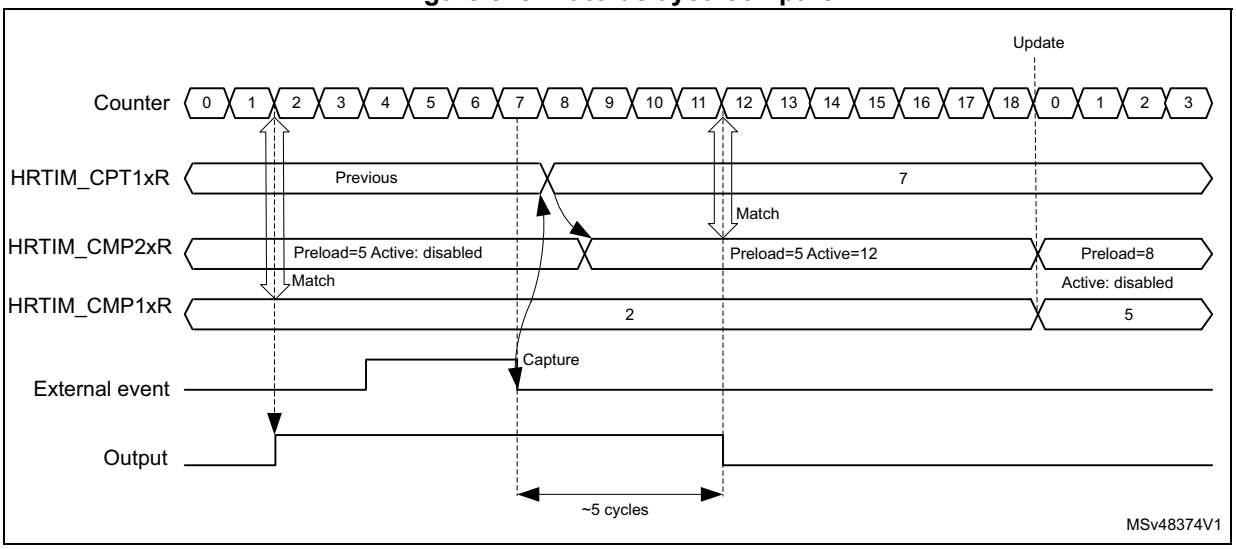
- DELCMPx[1:0] = 10
- DELCMPx[1:0] = 00
- DELCMPx[1:0] = 11

As an example, *Figure 348* shows how the following signal is generated:

- Output set when the counter is equal to compare 1 value
- Output reset 5 cycles after a falling edge on a given external event

Note: To simplify the figure, the high-resolution is not used in this example (CKPSC[2:0] = 101), thus the counter is incremented at the f_{HRTIM} rate. Similarly, the external event signal is shown without any resynchronization delay: practically, there is a delay of 1 to 2 f_{HRTIM} clock periods between the falling edge and the capture event due to an internal resynchronization stage which is necessary to process external input signals.

Figure 348. Auto-delayed compare



A regular compare channel (for example, compare 1) is used for the output set: as soon as the counter matches the content of the compare register, the output goes to its active state.

A delayed compare is used for the output reset: the compare event can be generated only if a capture event has occurred. No event is generated when the counter matches the delayed compare value (counter = 4). Once the capture event has been triggered by the external event, the content of the capture register is summed to the delayed compare value to have the new compare value. In the example, the auto-delayed value 4 is summed to the capture equal to 7 to give a value of 12 in the auto-delayed compare register. From this time on, the compare event can be generated and happens when the counter is equal to 12, causing the output to be reset.

Overcapture management in auto-delayed mode

Overcapture is prevented when the auto-delayed mode is enabled (DELCMPx[1:0] = 01, 10, 11).

When multiple capture requests occur within the same counting period, only the first capture is taken into account to compute the auto-delayed compare value. A new capture is possible only:

- Once the auto-delayed compare has matched the counter value (compare event)
- Once the counter has rolled over (period)
- Once the timer has been reset

Changing auto-delayed compare values

When the auto-delayed compare value is preloaded (PREEN bit set), the new compare value is taken into account on the next coming update event (for instance on the period event), regardless of when the compare register was written and if the capture occurred (refer to *Figure 348*, where the delay is changed when the counter rolls over).

When the preload is disabled (PREEN bit reset), the new compare value is taken into account immediately, even if it is modified after the capture event has occurred, as per the example below:

1. At t1, DELCMP2 = 1.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, a capture event occurs capturing the value CPTR1 = 0x20. => comparison enabled, compare value = 0x60
4. At t4, CMP2_act = 0x100 (before the counter reached value CPTR1 + 0x40) => comparison still enabled, new compare value = 0x120
5. At t5, the counter reaches the period value => comparison disabled, cmp2_act = 0x100

Similarly, if the CMP1(CMP3) value changes while DELCMPx = 10 or 11, and preload is disabled:

1. At t1, DELCMP2 = 2.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, CMP3 event occurs - CMP3_act = 0x50 before capture 1 event occurs => comparison enabled, compare value = 0x90
4. At t4, CMP3_act = 0x100 (before the counter reached value 0x90) => comparison still enabled, compare 2 event occurs at = 0x140

Triggered-half mode

The purpose of this mode is to allow the synchronization of 2 interleaved converters that have variable frequency operation and require a 180° phase-shift. The basic principle is to have a master-slave system.The slave converter synchronization is continuously adjusted based on the previous switching period of the master converter.

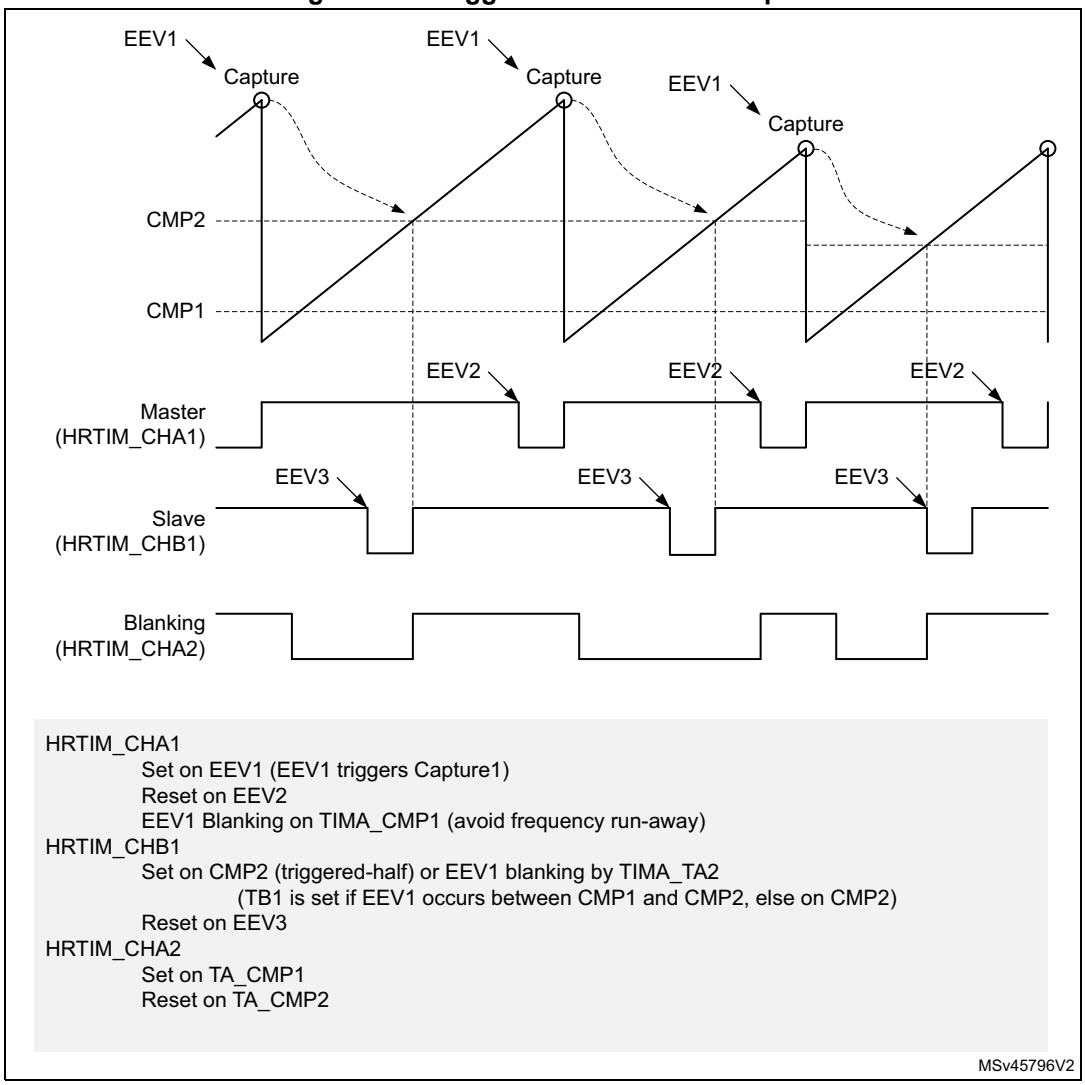
This is done using the capture unit. The switching period of the master converter is captured, divided by 2 and then stored in the compare 2 register by hardware. The compare 2 register contains a value equal to half of the captured period, which is the master converter switching period. The compare 2 event can then be used to trigger a second timing unit that manages the slave converter.

This mode is enabled by setting the TRGHLF bit in the HRTIM_TIMxCR2 register. This bit cannot be changed once the timer is operating (TxEN bit set).

The triggered-half mode must not be used simultaneously with other modes using CMP2 (dual channel dac trigger, interleaved and balanced idle modes).

The initial value CMP2 can be written by the user, but is ignored once the first capture is triggered. The preload mechanism is disabled for CMP2 when the TRGHLF bit is reset.

Figure 349. Triggered-half mode example



Note: In triggered half mode, the compare2 register is controlled by hardware and writing it has no effect. However the written value is stored in the preload register and becomes active on the update event following the exit of this mode.

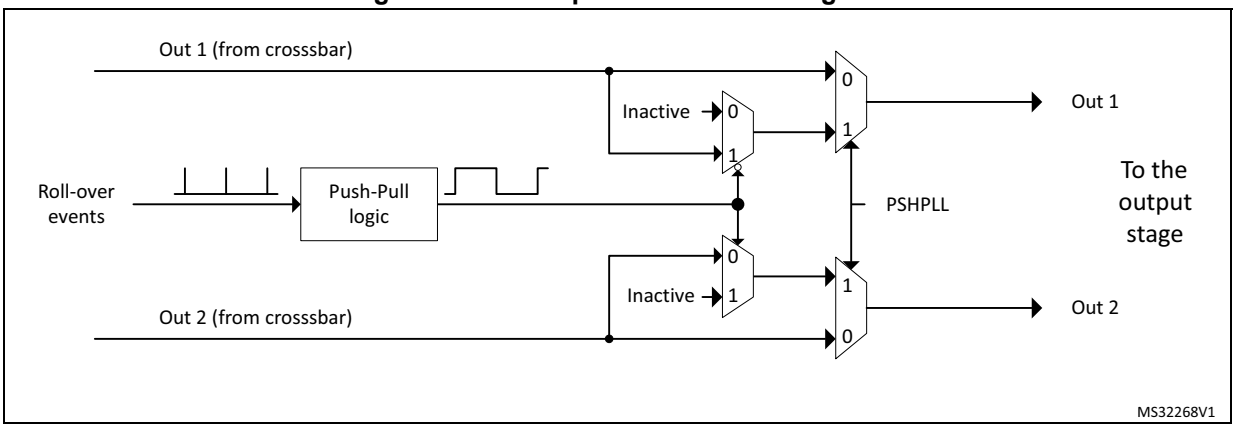
Push-pull mode

This mode primarily aims at driving converters using push-pull topologies. It also needs to be enabled when the delayed idle protection is required, typically for resonant converters (refer to *Section 34.3.10: Delayed protection*).

The push-pull mode is enabled by setting PSHPLL bit in the HRTIM_TIMxCR register.

It applies the signals generated by the crossbar to output 1 and output 2 alternatively, on the period basis, maintaining the other output to its inactive state. The redirection rate (push-pull frequency) is defined by the timer's period event, as shown in *Figure 350*. The push-pull period is twice the timer counting period.

Figure 350. Push-pull mode block diagram



The push-pull mode is available when the timer operates in continuous mode and in single-shot mode. It is necessary to disable the timer to stop a push-pull operation and to reset the counter before re-enabling it.

To get a correct behavior, the event selected as source of the counter reset must be also selected for setting (or resetting) the output. It must set the output if the output is set on period, else it must reset it. If it is not done, the output switching from its inactive period to its active period may be incorrect (may unexpectedly rise or may unexpectedly stay low).

The signal shape is defined using HRTIM_SETxyR and HRTIM_RSTxyR for both outputs. It is necessary to have HRTIM_SETx1R = HRTIM_SETx2R and HRTIM_RSTx1R = HRTIM_RSTx2R to have both outputs with identical waveforms and to achieve a balanced operation. Still, it is possible to have different programming on both outputs for other uses.

The CPPSAT status bit in HRTIM_TIMxISR indicates on which output the signal is currently active. CPPSTAT is reset when the push-pull mode is disabled.

In the example given in *Figure 351*, the timer internal waveform is defined as follows:

- Output set on period event
- Output reset on compare 1 match event

Figure 351. Push-pull mode example

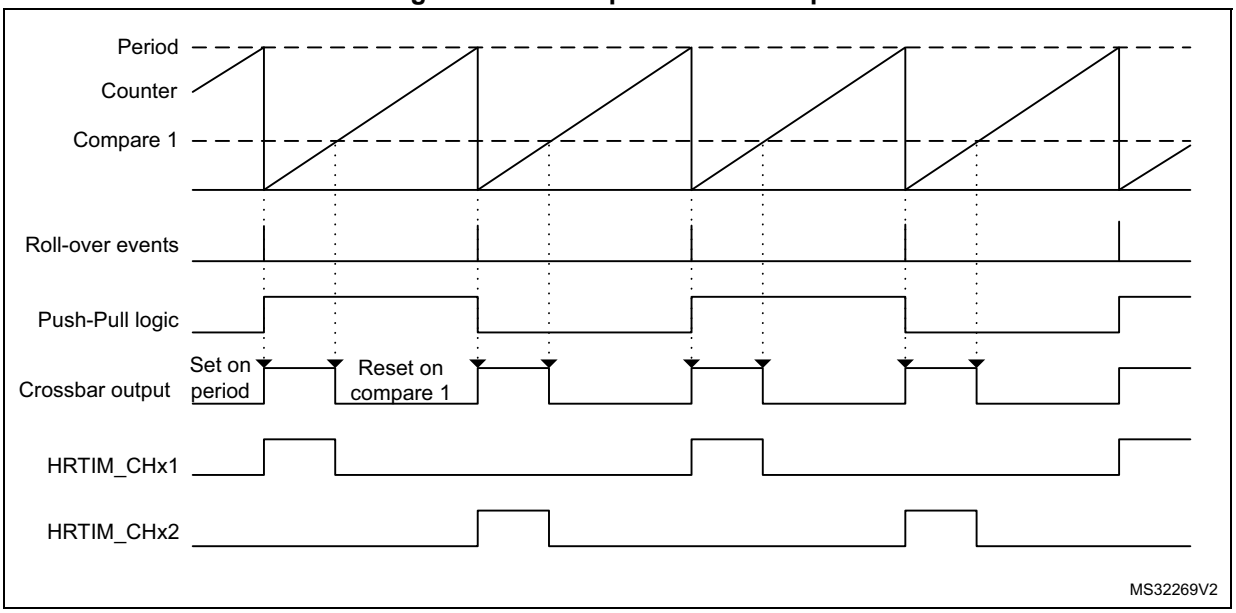
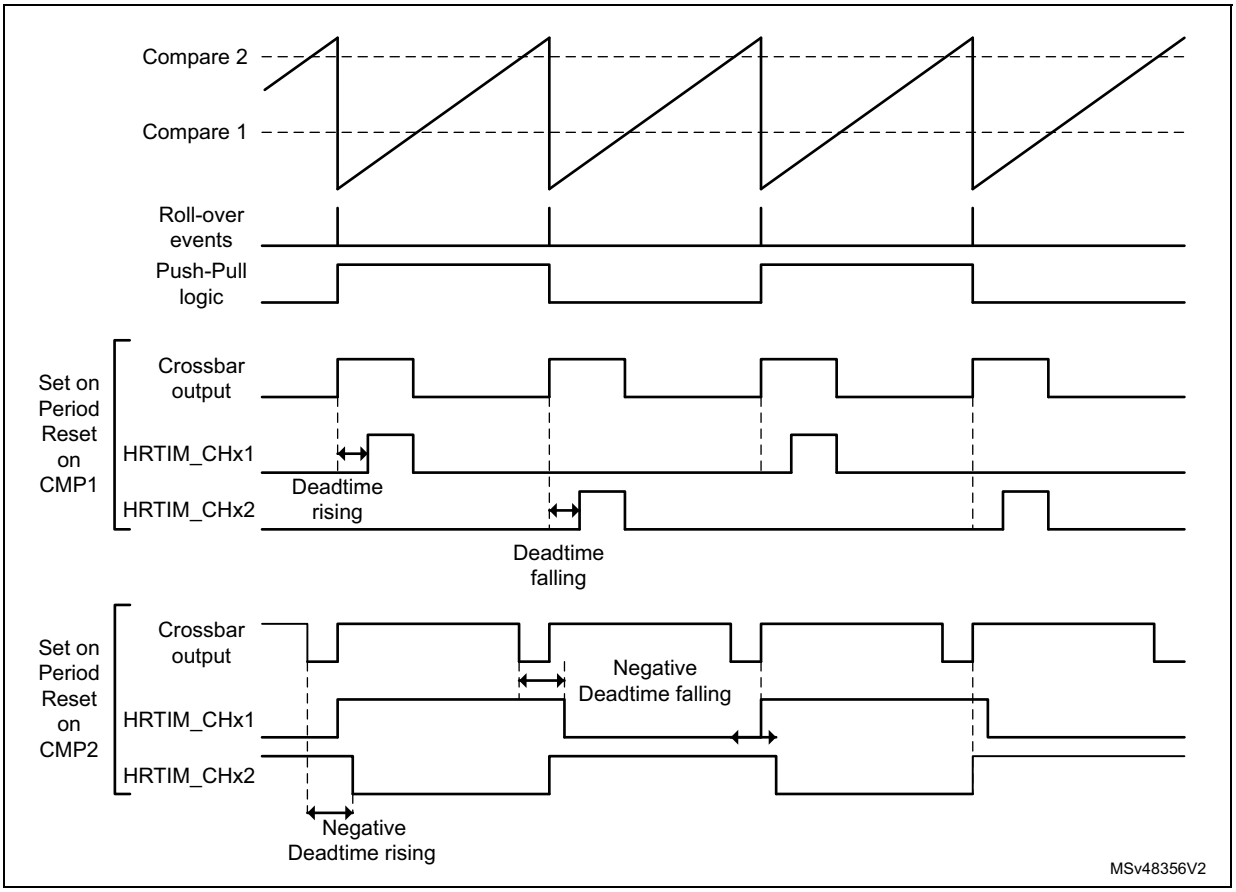


Figure 352 shows how the deadtime is inserted in push-pull mode, for both positive and negative deadtimes. In this case, the outputs are not complemented any more, and the deadtime is applied on each output individually (both ouput 1 and output 2 of the crossbar are used).

Figure 352. Push-Pull with deadtime



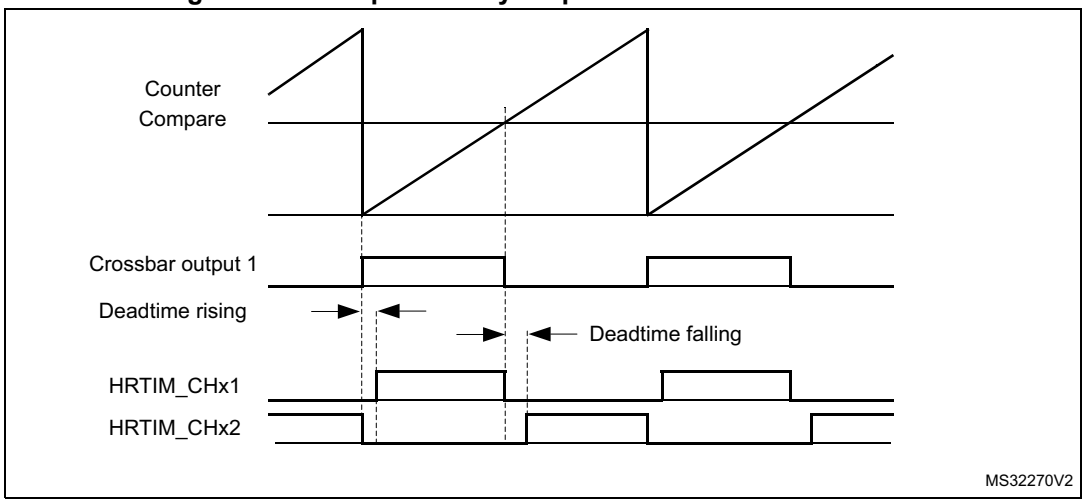
Deadtime

A deadtime insertion unit allows to generate a couple of complementary signals from a single reference waveform, with programmable delays between active state transitions. This is commonly used for topologies using half-bridges or full bridges. It simplifies the software: only 1 waveform is programmed and controlled to drive two outputs.

The Dead time insertion is enabled by setting DTEN bit in HRTIM_OUTxR register. The complementary signals are built based on the reference waveform defined for output 1, using HRTIM_SETx1R and HRTIM_RSTx1R registers: HRTIM_SETx2R and HRTIM_RSTx2R registers are not significant when DTEN bit is set.

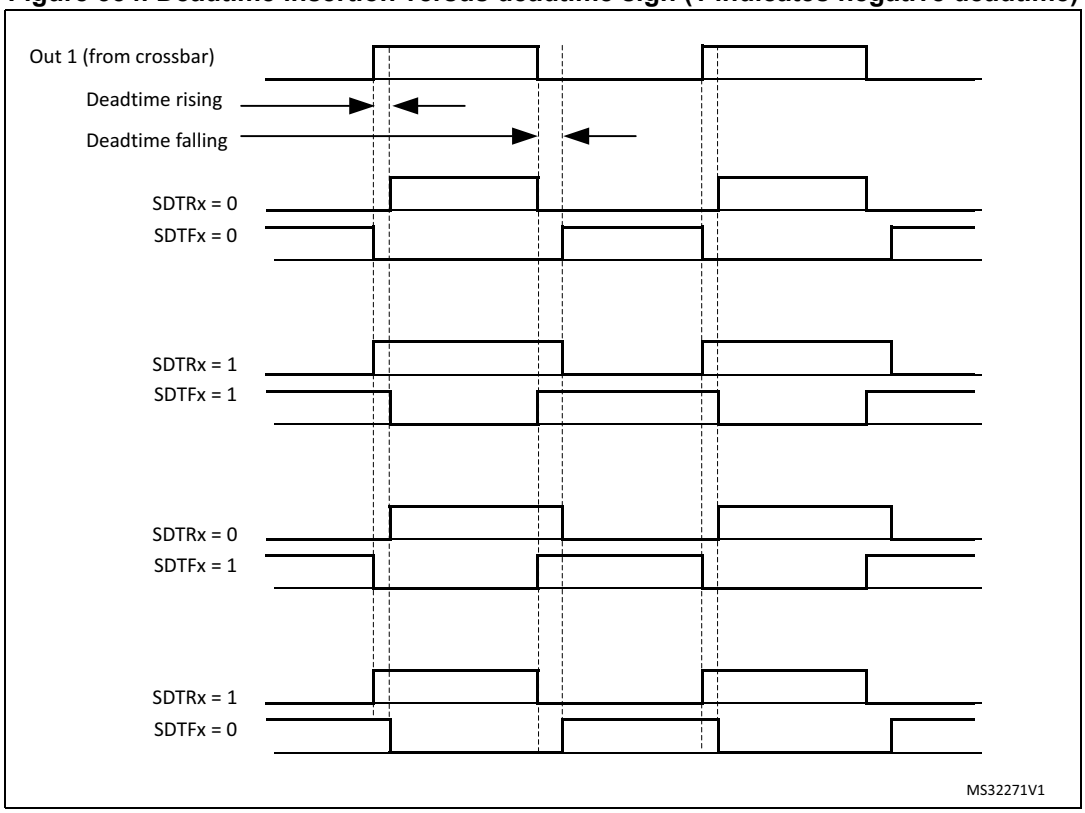
Two deadtimes can be defined in relationship with the rising edge and the falling edge of the reference waveform, as in *Figure 353*.

Figure 353. Complementary outputs with deadtime insertion



Negative deadtime values can be defined when some control overlap is required. This is done using the deadtime sign bits (SDTFx and SDTRx bits in HRTIM_DTxR register). *Figure 354* shows complementary signal waveforms depending on respective signs.

Figure 354. Deadtime insertion versus deadtime sign (1 indicates negative deadtime)



The deadtime values are defined with DTFx[8:0] and DTRx[8:0] bit fields and based on a specific clock prescaled according to DTPRSC[2:0] bits, as follows:

t_{DTx} = +/- DTx[8:0] x t_{DTG}

where x is either R or F and t_{DTG} = ($2^{(DTPRSC[2:0])}$) x (t_{HRTIM} /8).

Table 388 gives the resolution and maximum absolute values depending on the prescaler value.

Table 388. Deadtime resolution and max absolute values

DTPRSC[2:0]	t_{DTG}	t_{DTx} max	f_{HRTIM}= 300 MHz	
			t_{DTG} (ns)	$\|t_{DTx}\|$ max (µs)
000	t_{HRTIM} /8	511 * t_{DTG}	0.417	0.21
001	t_{HRTIM} /4		0.833	0.43
010	t_{HRTIM} /2		1.67	0.85
011	t_{HRTIM}		3.33	1.7
100	t_{HRTIM} * 2		6.67	3.41
101	t_{HRTIM} * 4		13.33	6.81
110	t_{HRTIM} * 8		26.67	13.63
111	t_{HRTIM} * 16		53.33	27.52

Figure 355 to *Figure 358* present how the deadtime generator behaves for reference waveforms with pulsewidth below the deadtime values, for all deadtime configurations.

Figure 355. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 0)

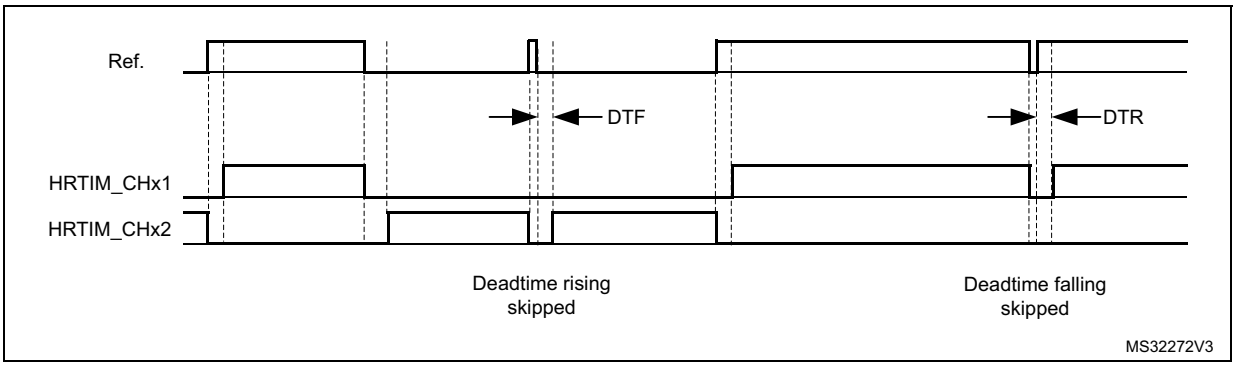


Figure 356. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 1)

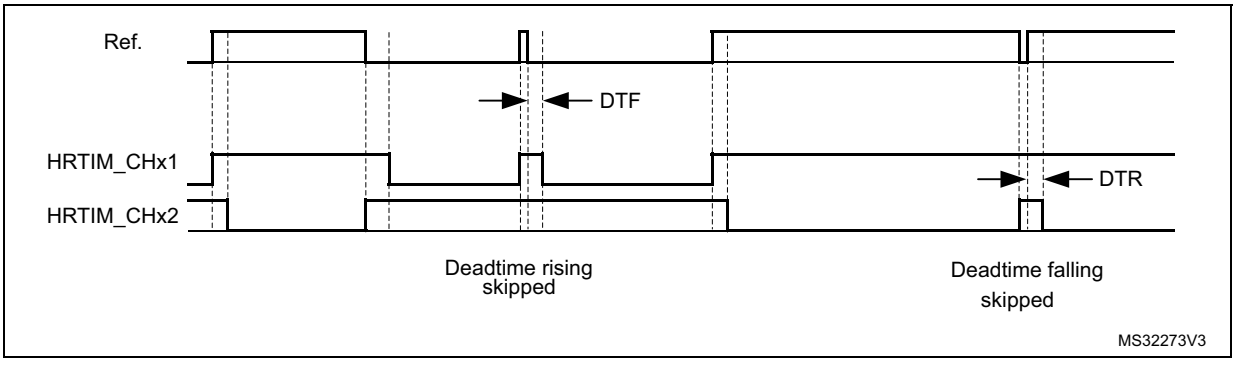


Figure 357. Complementary outputs for low pulsewidth (SDTRx = 0, SDTFx = 1)

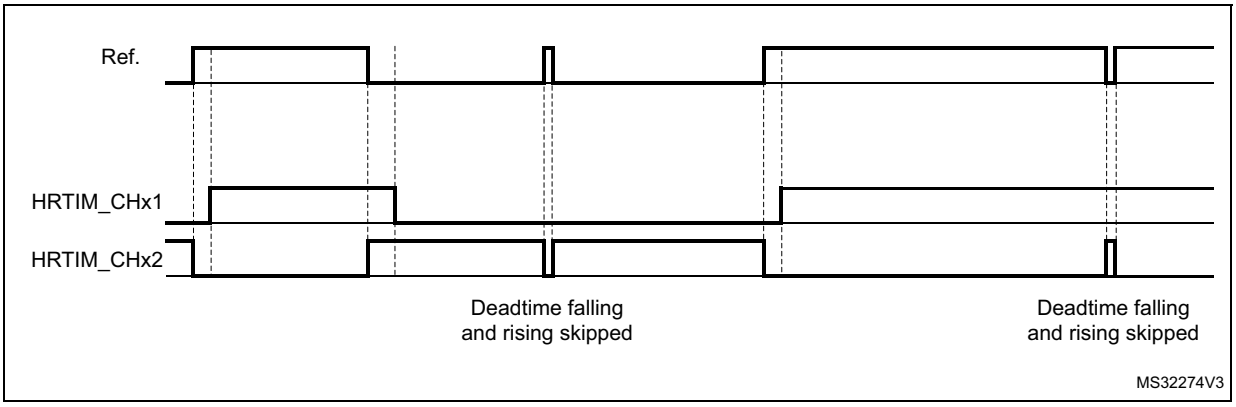
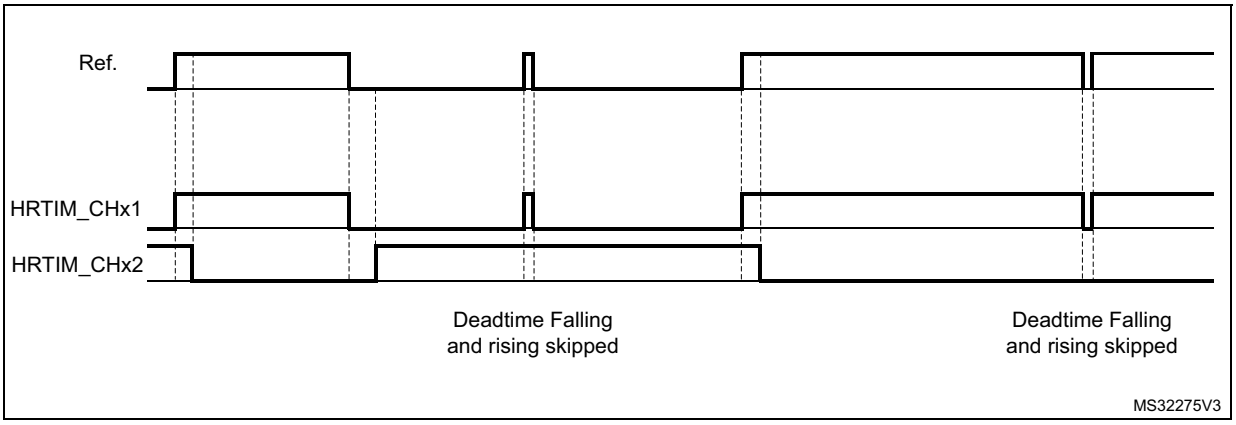


Figure 358. Complementary outputs for low pulsewidth (SDTRx = 1, SDTFx=0)



For safety purposes, it is possible to prevent any spurious write into the deadtime registers by locking the sign and/or the value of the deadtime using DTFLKx, DTRLKx, DTFSLKx and DTRSLKx. Once these bits are set, the related bits and bit fields are becoming read only until the next system or IP reset.

Caution: DTEN bit must not be changed in the following cases:
When the timer is enabled (TxEN bit set)
When the timer outputs are set/reset by another timer (while TxEN is reset)
Otherwise, an unpredictable behavior would result.
It is therefore necessary to disable the timer (TxCEN bit reset) and have the corresponding outputs disabled.

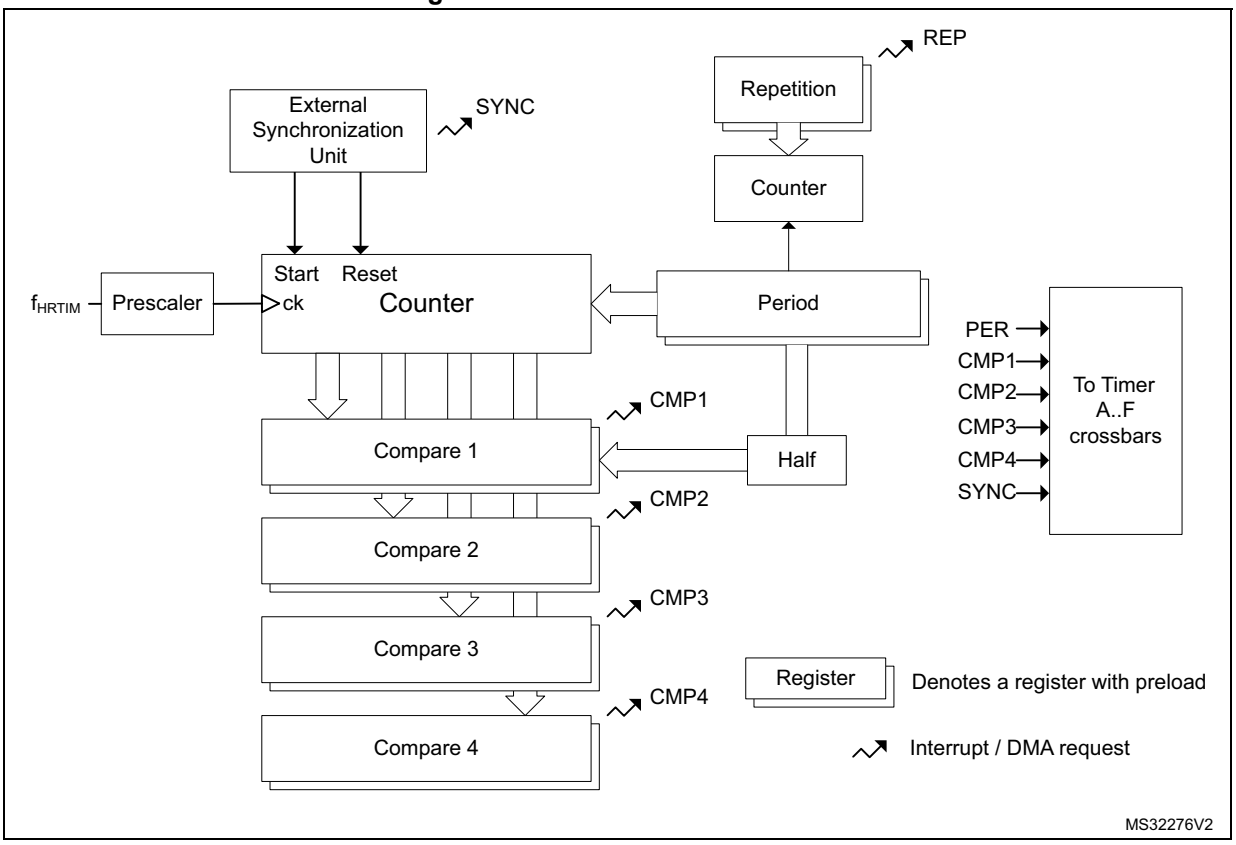
Note: For the particular case where DTEN must be set while the burst mode is enabled with a deadtime upon entry (BME = 1, DIDL = 1, IDLEM = 1), it is necessary to force the two outputs in their IDLES state by software commands (SST, RST bits) before setting DTEN bit. This is to avoid any side effect resulting from a burst mode entry that would happen immediately before a deadtime enable.

34.3.5 Master timer

The main purpose of the master timer is to provide common signals to the 6 timing units, either for synchronization purpose or to set/reset outputs. It does not have direct control over any outputs, but still can be used indirectly by the set/reset crossbars.

Figure 359 provides an overview of the master timer.

Figure 359. Master timer overview



The master timer is based on the very same architecture as the timing units, with the following differences:

- It does not have outputs associated with, nor output related control.
- It does not have its own crossbar unit, nor push-pull or deadtime mode.
- It can only be reset by the external synchronization circuitry.
- It does not have a capture unit, nor the auto-delayed mode.
- It does not include external event blanking and windowing circuitry.
- It has a limited set of interrupt / DMA requests: compare 1..4, repetition, register update and external synchronization event.

The master timer control register includes all the timer enable bits, for the master and timer A..F timing units. This allows to have all timers synchronously started with a single write access.

It also handles the external synchronization for the whole HRTIM timer (refer to *Section 34.3.19: Synchronizing the HRTIM with other timers or HRTIM instances*), with both MCU internal and external (inputs/outputs) resources.

Master timer control registers are mapped with the same offset as the timing units' registers.

34.3.6 Up-down counting mode

The HRTIM is natively designed with up-counters. It offers however an operating mode with up-down counters, also called center-aligned mode.

This mode is enabled using the UDM bit in the HRTIM_TIMxCR2 register. This bit must not be changed once the timer is operating (TxEN bit set). It is only available for the TIMA..F. The master timer only works in up-counting mode.

Not all HRTIM features are supported in up-down counting. This section details the functional differences versus up-counting mode.

The period in HRTIM_PERxR must be preloaded (or static) in up-down mode. It can be updated only on period event or on counter reset.

The set/reset crossbar programming differs as follows:

The events coming from the timing units are setting/resetting the outputs depending on the counter up/down direction:

- If the event is enabled in the HRTIM_SETxyR register, it sets the output during up-counting and resets it during down-counting.
- If the event is enabled in the HRTIM_RSTxyR register, it resets the output during up-counting and sets it during down-counting.
- If the events are enabled both in HRTIM_SETxyR and HRTIM_RSTxyR registers, the output toggles.

This applies to:

- The timing unit: period, compare 1..4, register update (6 events)
- The master timer: period, compare 1..4, HRTIM synchronization (6 events)
- All other timing units (for example, timer B..F for timer A): TIMEVNT1..9 (9 events described in *Table 385*)

Figure 360 below shows how to generate basic waveforms.

Figure 360. Basic symmetric waveform in up-down counting mode

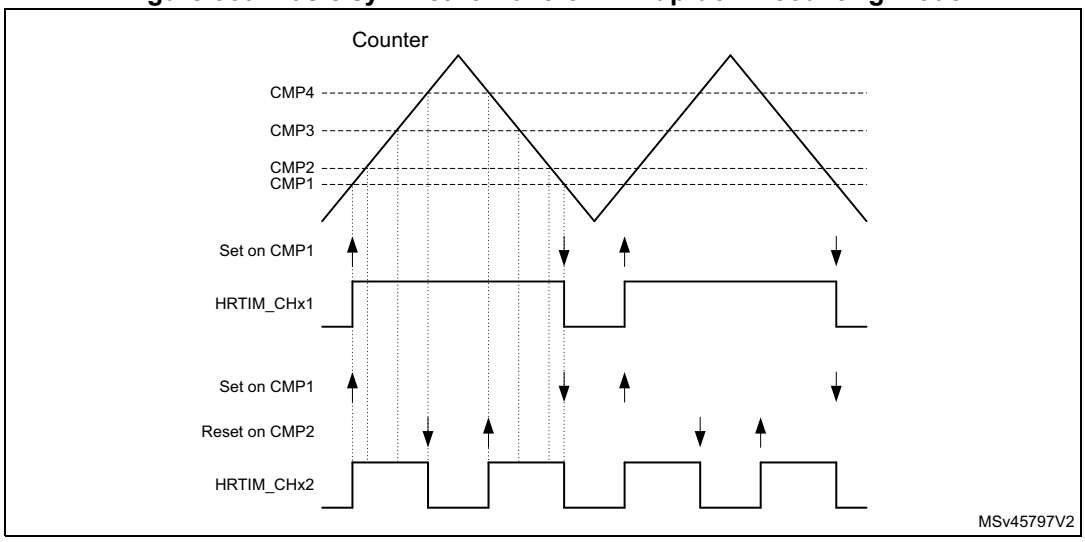


Figure 361 below shows how to generate some more complex waveforms, using the 4 available compare units and the toggle mode.

Figure 361. Complex symmetric waveform in up-down counting mode

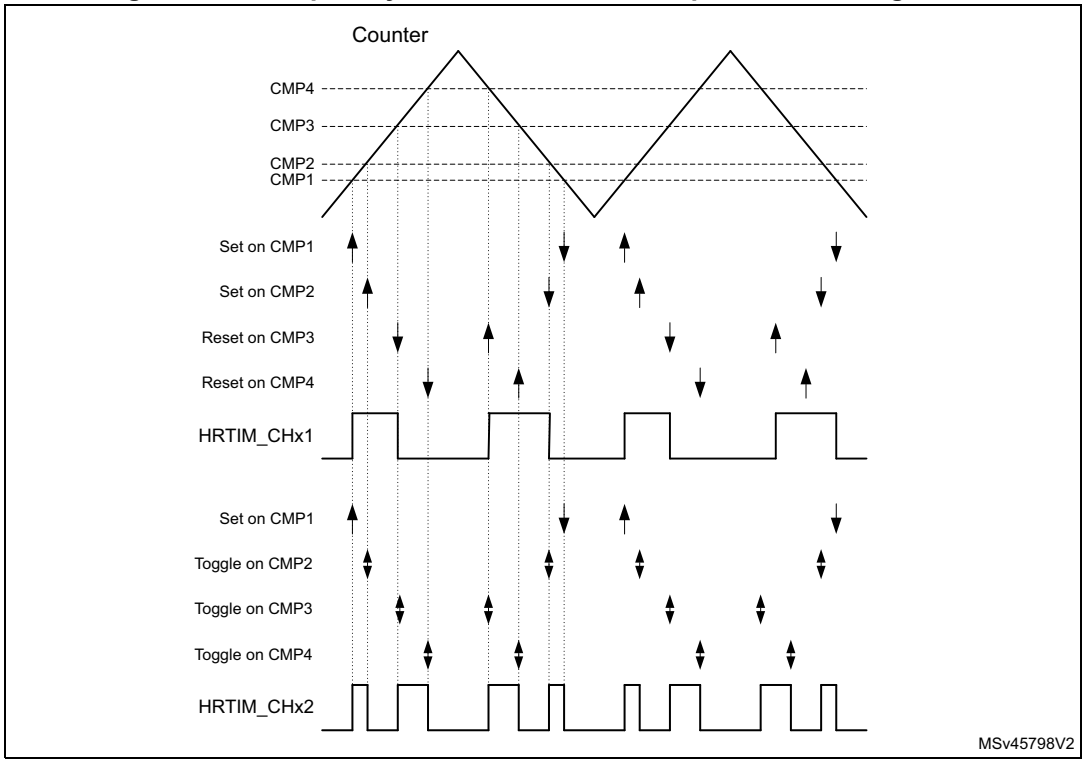
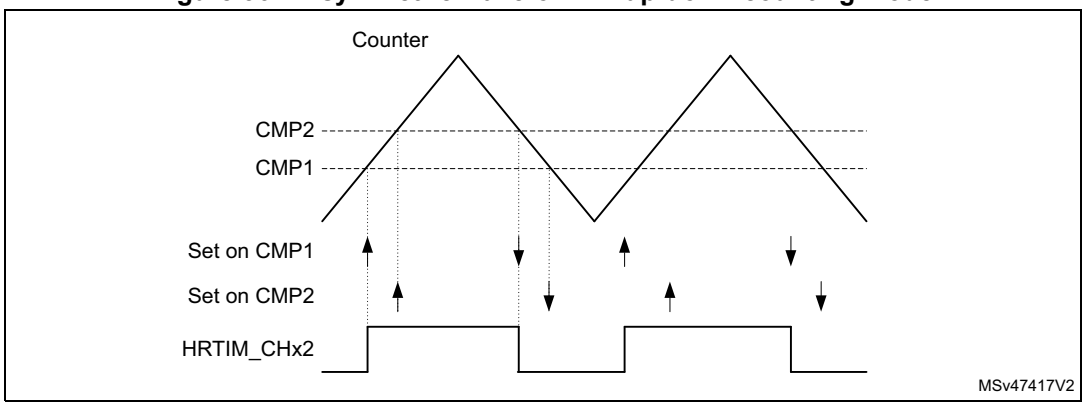


Figure 362 shows how to generate an asymmetric waveform. In this case, it must be noticed that it is necessary to have compare 2 value greater than compare 1 value for the waveform to be asymmetric.

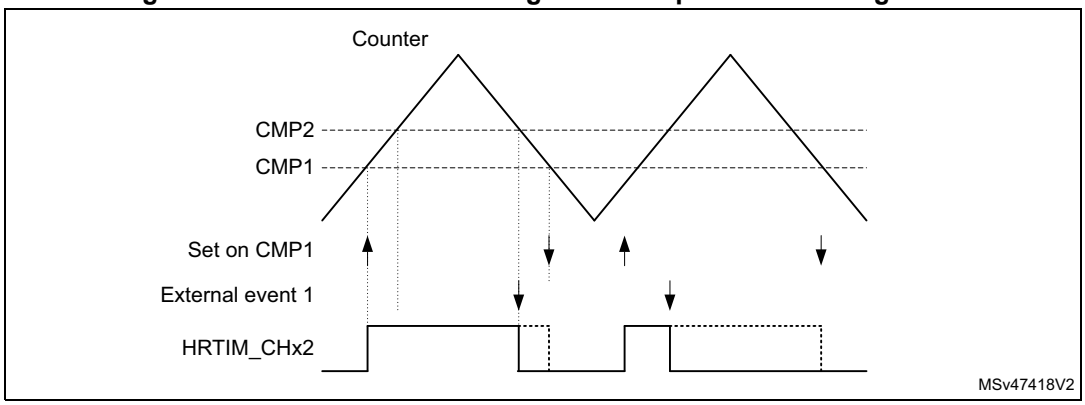
Figure 362. Asymmetric waveform in up-down counting mode



Note: For asymmetric operation, it is required that CMP2 > CMP1.

The behavior of the software forcing bits and external events EXTEVNT1..10 is identical for up-only and up-down counting mode. *Figure 363* shows how a pulse can be shortened in response to external events.

Figure 363. External event management in up-down counting mode



The up-down counting mode is available for both continuous and single-shot (retriggerable and non-retriggerable) operating modes. A reset causes the counter to re-start from 0. *Figure 364* shows the counter behavior in single-shot retriggerable mode, for TimB.

Figure 364. Interleaved up-down counter operation

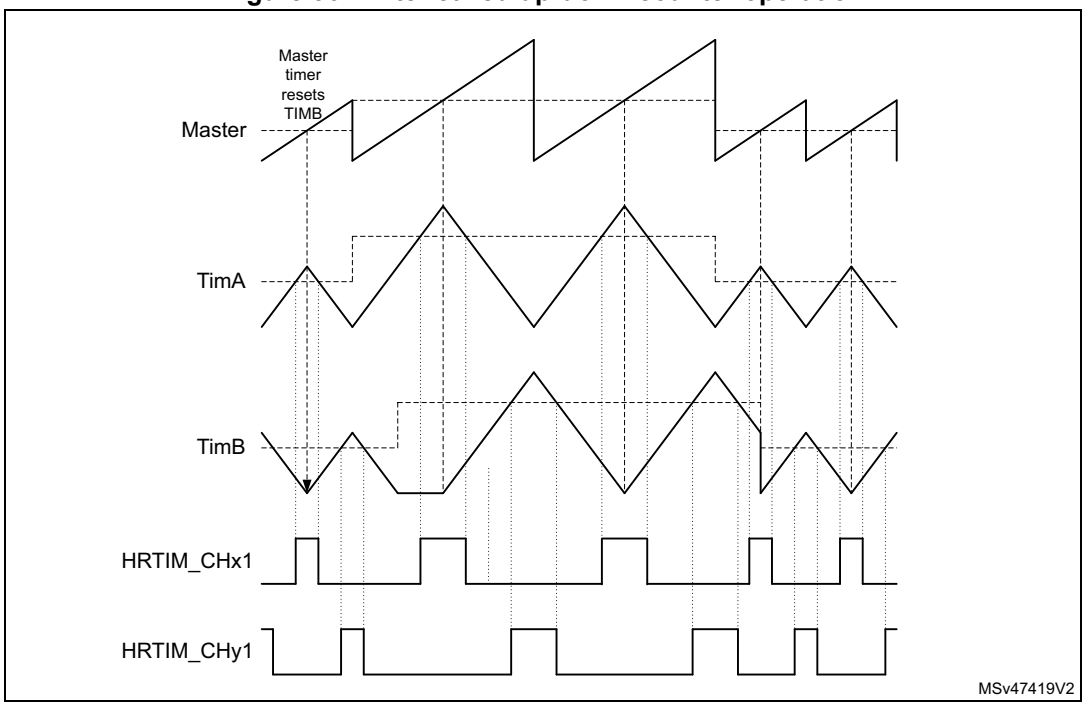
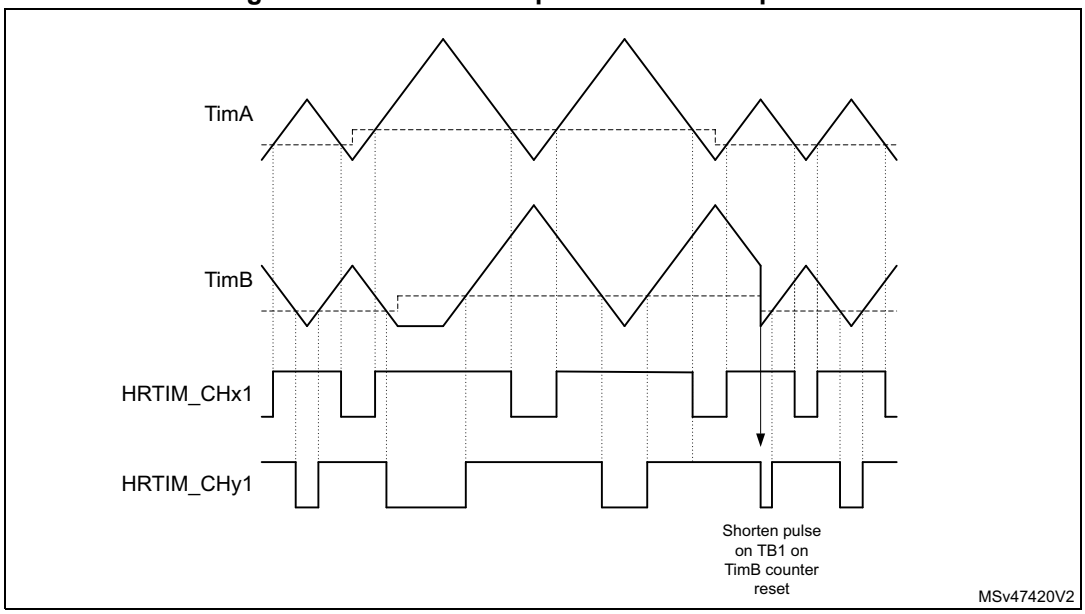


Figure 365. Interleaved up-down counter operation



Note: In up-down counting mode, the compare values must be 3 periods of the fHRTIM clock below the period value (TIMx_PER - 0xC0 if CKPSC[2:0] = 0, TIMx_PER - 0x60 if CKPSC[2:0] = 1, TIMx_PER - 0x30 if CKPSC[2:0] = 2,...). This applies for the compare events generated inside the timing unit. For compare events generated in other timing units, it is mandatory to avoid any event occurring within less than 1 period of the fHRTIM clock of a counter direction change (counter at 0, period event or counter reset).

The following features are supported in up-down counting mode:

- Half mode
- Deadtime insertion
- Push-pull mode, alternation push-pull done on when counter = 0 (refer to *Figure 366*)
- Delayed Idle
- Burst mode
- PWM mode with “greater than” comparison (refer to *Figure 367*)

Figure 366. Push-pull up-down mode example

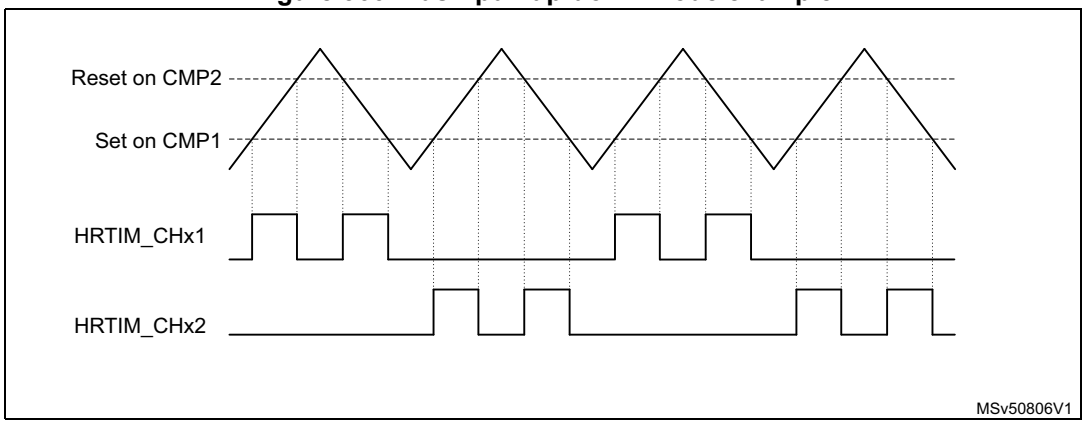
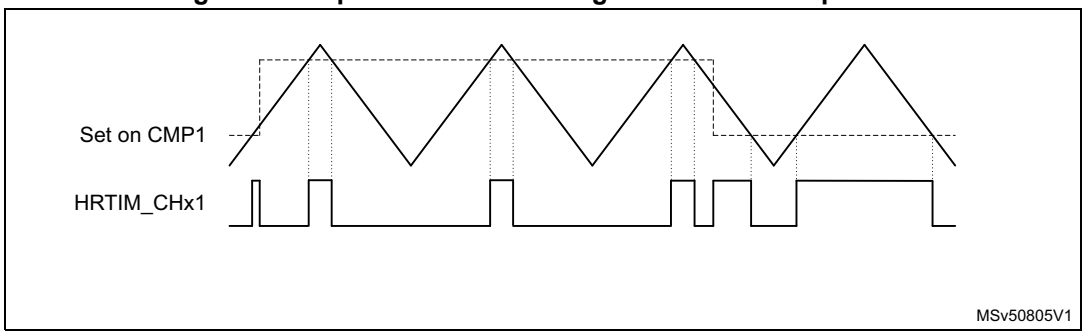


Figure 367. Up-down mode with “greater than” comparison



Caution: The following features are not supported in up-down counting mode:

- Auto-delayed mode
- Balanced Idle
- Triggered-half mode

The capture function is supported with the following differences:

- The capture value is referred to counting start, when up-counting
- The capture value is referred to the PER event when down counting
- The bit 16 of the capture register holds the counting direction status

The counter roll-over event is defined differently in up-down counting mode to support various operating conditions. It can be either generated:

- When the counter is equal to 0 (“valley” mode)
- When the counter reaches the period value set in the HRTIM_PERxR (“crest” mode)
- When both conditions are met (either 0 or HRTIM_PERxR value)

This event is used for multiple purposes in the HRTIM. The generation mode (valley, crest or both) can be programmed individually depending on the destination. *Table 389* summarizes the use cases and associated roll-over mode (xxROM[1:0]) programming bits in the HRTIM_TIMxCR2 register.

Table 389. Roll-over event destination and mode programming

Roll-over event use	Programming bits
Output set/reset	OUTROM[1:0]
Register content update trigger (transfer from preload to active)	ROM[1:0]
IRQ and/or DMA request trigger	ROM[1:0]
Burst mode clock source and /or burst start trigger	BMROM[1:0]
ADC trigger (refer to *ADC post-scaler* for details)	ADROM[1:0]
External event filtering	ROM[1:0]
Repetition counter decrement	ROM[1:0]
Fault and event counter	FEROM[1:0]

Note: For events where both reset and roll-over are considered (IRQ/DMA and TxRSTU), the ROM[1:0] only influences the roll-over generation. The reset event is always taken into account whatever the ROM[1:0] value.

The roll-over event generation is defined as per the following xxROM[1:0] bit field setting:

- xxROM[1:0] = 00: event generated when both conditions are met (either 0 or HRTIM_PERxR value)
- xxROM[1:0] = 01: event generated when the counter is equal to 0 (“valley” mode) or when the counter is reset
- xxROM[1:0] = 10: event generated when the counter reaches the period value set in the HRTIM_PERxR (“crest” mode)

The *Figure 368* below shows a push-pull up-down mode with set on period event and OUTROM[1:0] =10.

Figure 368. Up-down mode with output set on period event, for OUTROM[1:0]=10

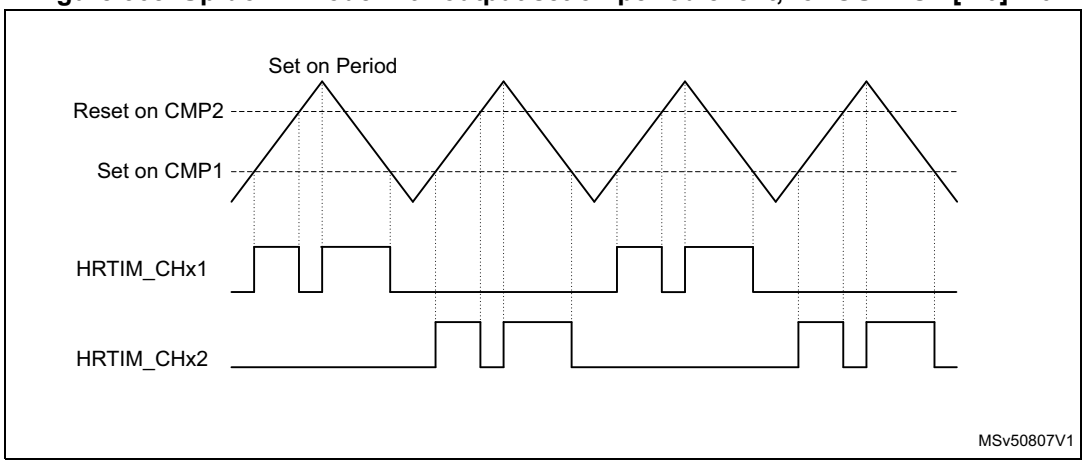
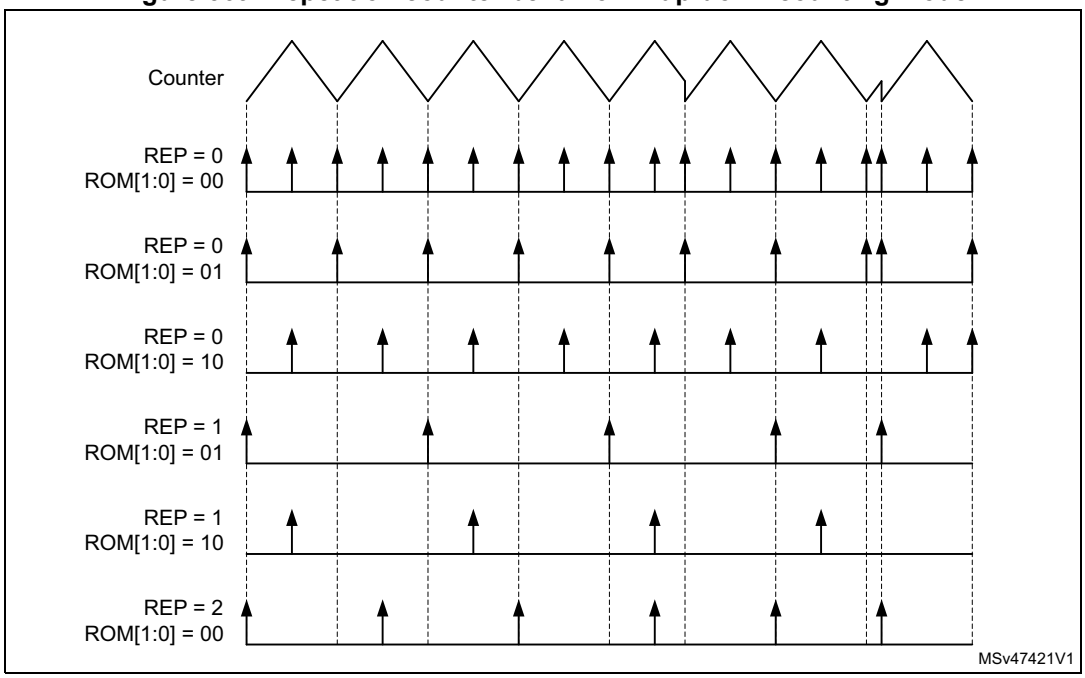


Figure 369 shows how the repetition counter is decremented in up-down counting mode.

Figure 369. Repetition counter behavior in up-down counting mode



The dual channel DAC triggers are working as for the up-counting mode.

The event blanking and windowing differs so as to have the blanking or windowing done within the output pulse, at a programmable time. The EExFLTR[3:0] codes are depending on the UDM bit setting, as per *Table 390*. Whenever the roll-over event is used for blanking or windowing, the ROM[1:0] programming applies for defining when it is generated.

Table 390. EExFLTR[3:0] codes depending on UDM bit setting

EE1FLTR[3:0]	Up-counting mode (UDM = 0)	Up/down-counting mode (UDM = 1)
0010	Blanking from counter reset/roll-over to compare 2	Blanking from compare 1 to compare 2, only during the up-counting phase
0100	Blanking from counter reset/roll-over to compare 4	Blanking from compare 3 to compare 4, only during the up-counting phase
1101	Windowing from counter reset/roll-over to compare 2	Windowing from compare 2 to compare 3, only during the up-counting phase
1110	Windowing from counter reset/roll-over to compare 3	Windowing from compare 2 to compare 3, only during the down-counting phase
1111	Windowing from another timing unit: TIMWIN source	Windowing from compare 2 during the up-counting phase to compare 3 during the down-counting phase

34.3.7 Set/reset event priorities and narrow pulse management

This section describes how the output waveform is generated when several set and/or reset requests are occurring within 3 consecutive t_{HRTIM} periods.

Case 1: clock prescaler CKPSC[2:0] < 5

An arbitration is performed during each t_{HRTIM} period, in 3 steps:

1. For each active event, the desired output transition is determined (set, reset or toggle).
2. A predefined arbitration is performed among the active events (from highest to lowest priority CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1 ⟶ PER, refer to *Concurrent set requests/ Concurrent reset requests*).
3. A high-resolution delay-based arbitration is performed with reset having the highest priority, among the low-resolution events and events having won the predefined arbitration.

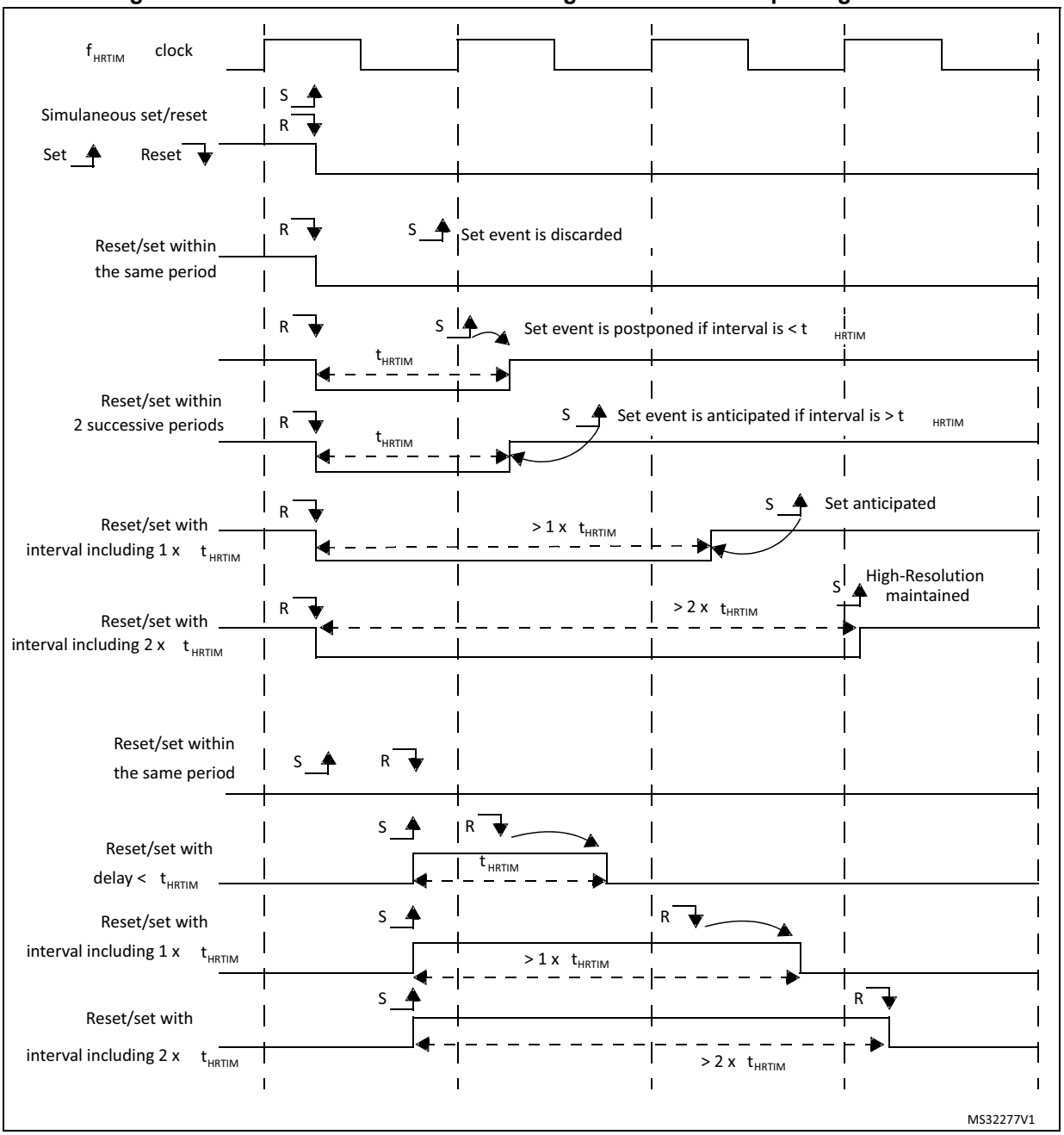
When set and reset requests from two different sources are simultaneous, the reset action has the highest priority. If the interval between set and reset requests is below 2 f_{HRTIM} period, the behavior depends on the time interval and on the alignment with the f_{HRTIM} clock, as shown in *Figure 370*.

Note: If the set and reset requests are simultaneous and coming from the same timing unit, the CMPx priority applies, as shown in step 2 here-above. For instance, taking CMP2 = CMP4:

If CMP2 does a set and CMP4 a reset, the output is reset,

If CMP2 does a reset and CMP4 a set, the output is set.

Figure 370. Short distance set/reset management for narrow pulse generation



If the set and reset events are generated within the same t_{HRTIM} period, the reset event has the highest priority and the set event is ignored.

If the set and reset events are generated with an interval below t_{HRTIM} period, across 2 periods, a pulse of 1 t_{HRTIM} period is generated.

If the set and reset events are generated with an interval below 2 t_{HRTIM} periods, a pulse of 2 t_{HRTIM} periods is generated.

If the set and reset events are generated with an interval between 2 and 3 t_{HRTIM} periods, the high-resolution is available if the interval is over 2 complete t_{HRTIM} periods.

If the set and reset events are generated with an interval above 3 t_{HRTIM} periods, the high-resolution is always available.

Concurrent set requests/ Concurrent reset requests

When multiple sources are selected for a set event, an arbitration is performed when the set requests occur within the same f_{HRTIM} clock period.

In case of multiple requests from adjacent timers (TIMEVNT1..9), the request which occurs first is taken into account. The arbitration is done in 2 steps, depending on:

1. The source (CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1),
2. The delay.

If multiple requests from the master timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and a single request is taken into account, whatever the effective high-resolution setting (from the highest to the lowest priority):

MSTCMP4 → MSTCMP3 → MSTCMP2 → MSTCMP1 → MSTCMPER

Note: It is advised to avoid generating multiple set (reset) requests from the master timer to a given timer with an interval below 3x t_{HRTIM} to maintain the high-resolution.

When multiple requests internal to the timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and the requests are taken with the following priority, whatever the effective timing (from highest to lowest) is:

CMP4 → CMP3 → CMP2 → CMP1 → PER

Note: Practically, this is of a primary importance when multiple compare events can be simultaneously generated or when using auto-delayed compare 2 and compare 4 simultaneously (that is when the effective set/reset cannot be determined a priori because it is related to an external event). In this case, the highest priority signal must be affected to the CMP4 event.

Last, the highest priority is given to low-resolution events: EXTEVNT1..10, RESYNC (coming from SYNC event if SYNCRSTx or SYNCSTRTx is set or from a software reset), update and software set (SST). The update event is considered as having the largest delay (0x1F if PSC = 0).

As a summary, in case of a close vicinity (events occurring within the same f_{HRTIM} clock period), the effective set (reset) event is arbitrated between:

- Any TIMEVNT1..9 event
- A single source from the master (as per the fixed arbitration given above)
- A single source from the timer
- The “low-resolution events”.

The same arbitration principle applies for concurrent reset requests. In this case, the reset request has the highest priority.

Case 2: clock prescaler CKPSC[2:0] ≥ 5

The narrow pulse management is simplified when the high-resolution is not effective.

A set or reset event occurring within the prescaler clock cycle is delayed to the next active edge of the prescaled clock (as for a counter reset), even if the arbitration is still performed every t_{HRTIM} cycle.

If a reset event is followed by a set event within the same prescaler clock cycle, the latest event is considered.

34.3.8 External events global conditioning

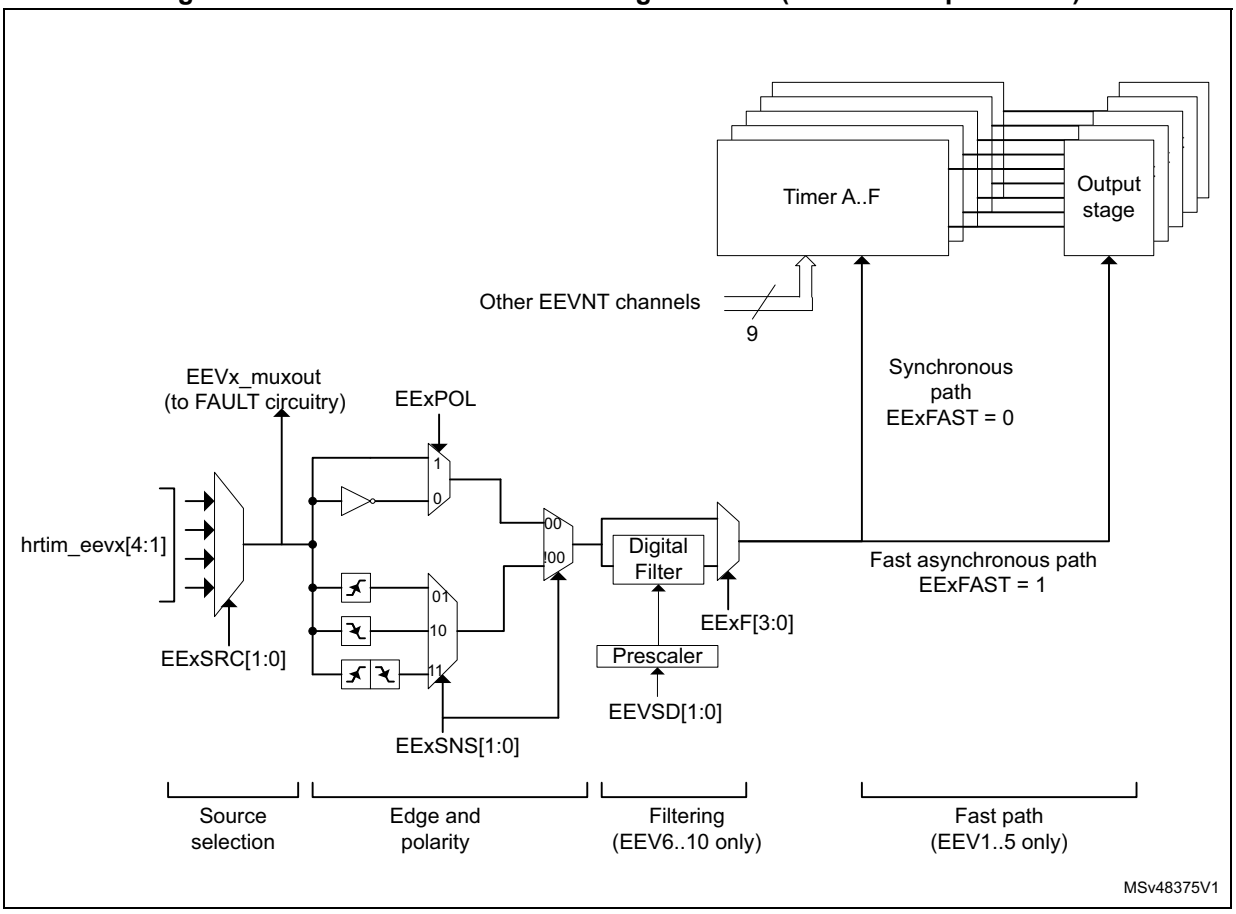
The HRTIM timer can handle events not generated within the timer, referred to as “external event”. These external events come from multiple sources, either on-chip or off-chip:

- Built-in comparators,
- Digital input pins (typically connected to off-chip comparators and zero-crossing detectors),
- On-chip events for other peripheral (ADC’s analog watchdogs and general purpose timer trigger outputs).

The external events conditioning circuitry allows to select the signal source for a given channel (with a 4:1 multiplexer) and to convert it into information that can be processed by the crossbar unit (for instance, to have an output reset triggered by a falling edge detection on an external event channel).

Up to 10 external event channels can be conditioned and are available simultaneously for any of the 5 timers. This conditioning is common to all timers, since this is usually dictated by external components (such as a zero-crossing detector) and environmental conditions (typically the filter set-up is related to the applications noise level and signature). *Figure 371* presents an overview of the conditioning logic for a single channel.

Figure 371. External event conditioning overview (1 channel represented)



The 10 external events are initialized using the HRTIM_EECR1 and HRTIM EECR2 registers:

- To select up to 4 sources with the EExSRC[1:0] bits,
- To select the sensitivity with EExSNS[1:0] bits, to be either level-sensitive or edge-sensitive (rising, falling or both),
- To select the polarity, in case of a level sensitivity, with EExPOL bit,
- To have a low latency mode, with EExFAST bits (refer to *Latency to external events*), for external events 1 to 5.

Note: The external events used as triggers for reset, capture, burst mode, ADC triggers and delayed protection are edge-sensitive even if EESNS bit is reset (level-sensitive selection): if POL = 0 the trigger is active on external event rising edge, while if POL = 1 the trigger is active on external event falling edge.

The external events are discarded as long as the counters are disabled (TxCEN bit reset) to prevent any output state change and counter reset, except if they are used as ADC triggers.

Additionally, it is possible to enable digital noise filters, for external events 6 to 10, using EExF[3:0] bits in the HRTIM_EECR3 register.

A digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and

transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected).

The sampling clock is either the f_{HRTIM} clock or a specific prescaled clock f_{EEVS} derived from f_{HRTIM}, defined with EEVSD[1:0] bits in HRTIM_EECR3 register.

Table 391 summarizes the available features associated with each of the 10 external events channels.The features and sources are summarized in HRTIM external events mapping and associated features tables in *Chapter 5: Device configuration*.

Table 391. External events features

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E,F
hrtim_eev1[4:1]	Yes	—	—	—
hrtim_eev2[4:1]	Yes	—	—	—
hrtim_eev3[4:1]	Yes	—	—	—
hrtim_eev3[4:1]	Yes	—	—	—
hrtim_eev5[4:1]	Yes	—	—	—
hrtim_eev6[4:1]	—	Yes	Yes	—
hrtim_eev7[4:1]	—	Yes	Yes	—
hrtim_eev8[4:1]	—	Yes	—	Yes
hrtim_eev9[4:1]	—	Yes	—	Yes
hrtim_eev10[4:1]	—	Yes	—	—

Latency to external events

The external event conditioning gives the possibility to adjust the external event processing time (and associated latency) depending on performance expectations:

- A regular operating mode, in which the external event is resampled with the clock before acting on the output crossbar. This adds some latency but gives access to all crossbar functionalities. It enables the generation of an externally triggered high-resolution pulse.
- A fast operating mode, in which the latency between the external event and the action on the output is minimized. This mode is convenient for ultra-fast over-current protections, for instance.

EExFAST bits in the HRTIM_EECR1 register allow to define the operating for channels 1 to 5. This influences the latency and the jitter present on the output pulses, as summarized in the table below.

Table 392. Output set/reset latency and jitter versus external event operating mode

EExFAST	Response time latency	Response time jitter	Jitter on output pulse (counter reset by ext. event)
0	5 to 6 cycles of f_{HRTIM} clock	1 cycles of f_{HRTIM} clock	No jitter, pulse width maintained with high-resolution
1	Minimal latency (depends whether the comparator or digital input is used)	Minimal jitter	1 cycle of f_{HRTIM} clock jitter pulse width resolution down to t_{HRTIM}

The EExFAST mode is only available with level-sensitive programming (EExSNS[1:0] = 00); the edge-sensitivity cannot be programmed.

It is possible to apply event filtering to external events (both blanking and windowing with EExFLTR[3:0]! = 0000, refer to *Section 34.3.9*). In this case, EExLTCHx bit must be reset: the postponed mode is not supported, neither the windowing timeout feature.

Note: The external event configuration (source and polarity) must not be modified once the related EExFAST bit is set.

A fast external event cannot be used to toggle an output: it must be enabled either in HRTIM_SETxyR or HRTIM_RSTxyR registers, not in both.

When a set and a reset event - from 2 independent fast external events - occur simultaneously, the reset has the highest priority in the crossbar and the output becomes inactive.

When EExFAST bit is set, the output cannot be changed during the 11 f_{HRTIM} clock periods following the external event.

Figure 372 and *Figure 373* give practical examples of the reaction time to external events, for output set/reset and counter reset.

Figure 372. Latency to external events falling edge (counter reset and output set)

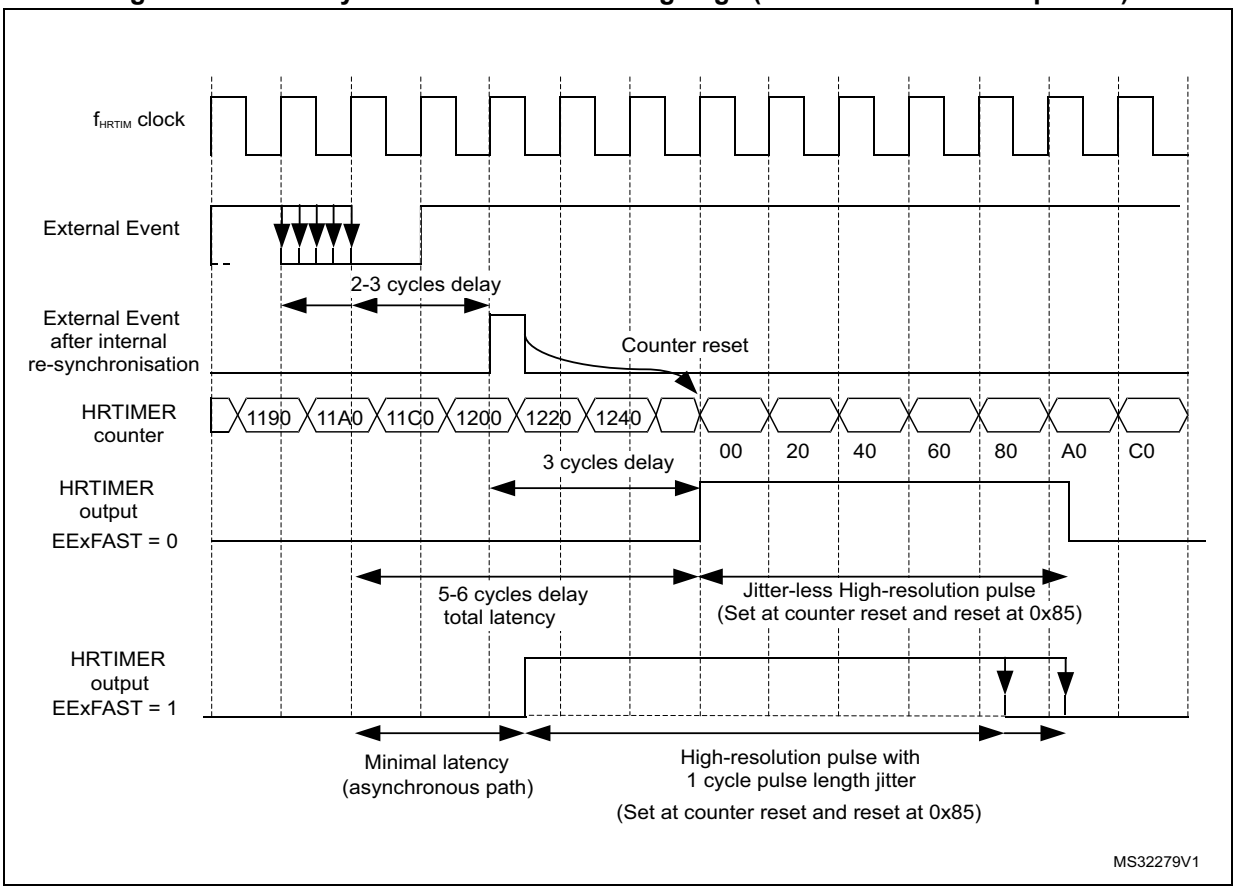
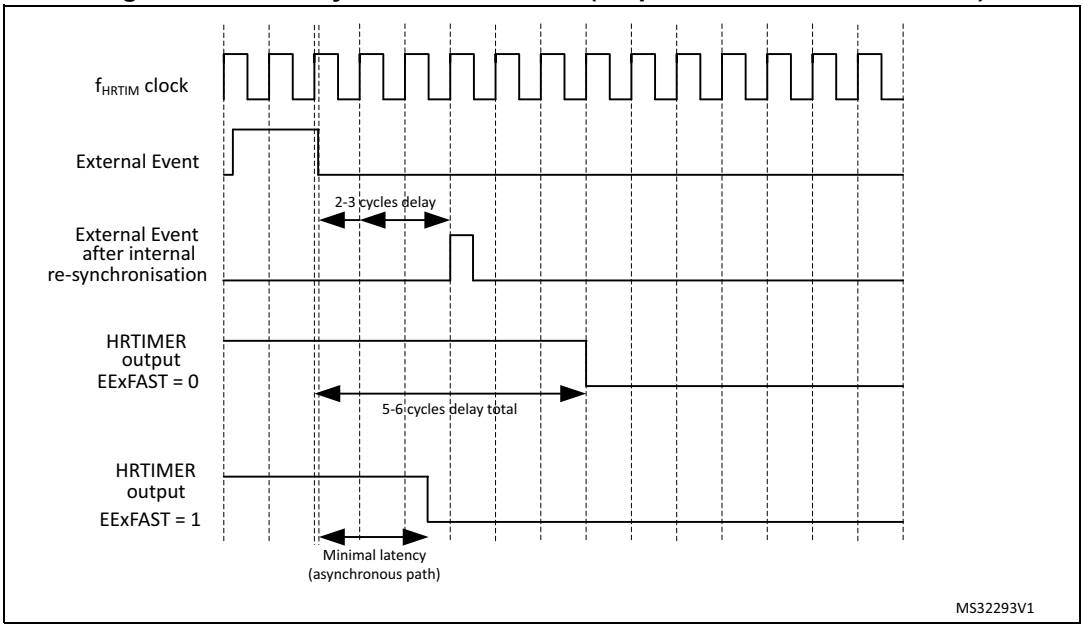


Figure 373. Latency to external events (output reset on external event)



34.3.9 External event filtering in timing units

Once conditioned, the 10 external events are available for all timing units.

They can be used directly and are active as soon as the timing unit counter is enabled (TxCEN bit set).

They can also be filtered to have an action limited in time, usually related to the counting period. Two operations can be performed:

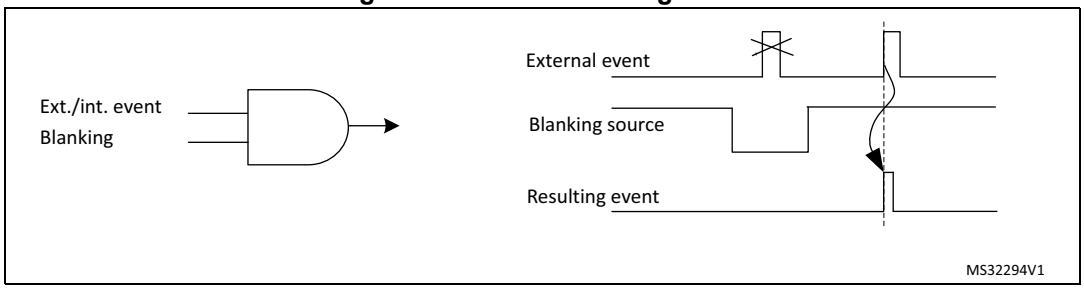
- Blanking, to mask external events during a defined time period,
- Windowing, to enable external events only during a defined time period.

These modes are enabled using HRTIM_EExFLTR[3:0] bits in the HRTIM_EEFxR1 and HRTIM_EEFxR2 registers. Each of the 6 timer A..F timing units has its own programmable filter settings for the 10 external events.

Blanking mode

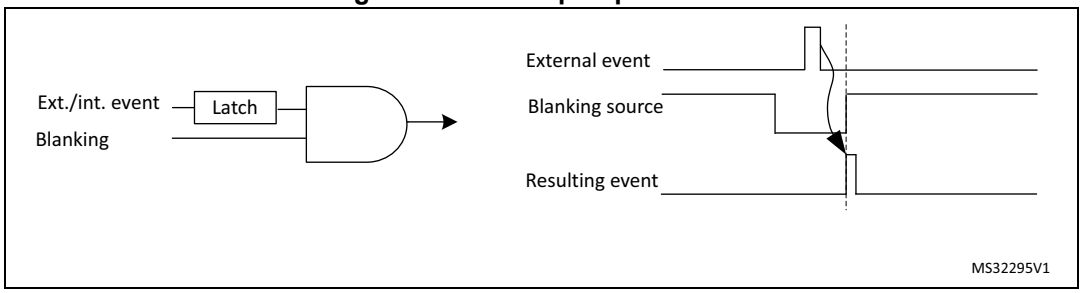
In event blanking mode (refer to *Figure 374*), the external event is ignored if it happens during a given blanking period. This is convenient, for instance, to avoid a current limit to trip on switching noise at the beginning of a PWM period. This mode is active for EExFLTR[3:0] bit field values ranging from 0001 to 1100.

Figure 374. Event blanking mode



In event postpone mode, the external event is not taken into account immediately but is memorized (latched) and generated as soon as the blanking period is completed, as shown in *Figure 375*. This mode is enabled by setting EExLTCH bit in HRTIM_EEFxR1 and HRTIM_EEFxR2 registers.

Figure 375. Event postpone mode



The blanking signal comes from several sources:

- The timer itself: the blanking lasts from the counter reset to the compare match (EExFLTR[3:0] = 0001 to 0100 for compare 1 to compare 4). In up/down mode (UDM bit set to 1), the counter reset event is defined as per the ROM[1:0] bit setting.
- From other timing units (EExFLTR[3:0] = 0101 to 1100): the blanking lasts from the selected timing unit counter reset to one of its compare match, or can be fully programmed as a waveform on Tx2 output. In this case, events are masked as long as the Tx2 signal is inactive (it is not necessary to have the output enabled, the signal is taken prior to the output stage).

The EEXFLTR[3:0] configurations from 0101 to 1100 are referred to as TIMFLTR1 to TIMFLTR8 in the bit description, and differ from one timing unit to the other. *Table 393* gives the 8 available options per timer: CMPx refers to blanking from counter reset to compare match, Tx2 refers to the timing unit TIMx output 2 waveform defined with HRTIM_SETx2 and HRTIM_RSTx2 registers. For instance, timer B (TIMFLTR6) is timer C output 2 waveform.

Table 393. Filtering signals mapping per timer

Source		Timer A				Timer B				Timer C				Timer D				Timer E				Timer F			
		CMP1	CMP2	CMP4	TA2	CMP1	CMP2	CMP4	TB2	CMP1	CMP2	CMP4	TC2	CMP1	CMP2	CMP4	TD2	CMP1	CMP2	CMP4	TE2	CMP1	CMP2	CMP4	TF2
Destination	Timer A	-	-	-	-	1	-	2	3	4	-	5	-	7	-	-	-	-	8	-	-	6	-	-	-
	Timer B	1	-	2	3	-	-	-	-	4	5	-	-	-	7	-	-	8	-	-	-	-	6	-	-
	Timer C	-	1	-	-	2	-	3	-	-	-	-	-	5	-	6	7	-	-	8	-	4	-	-	-
	Timer D	1	-	-	-	-	2	-	-	3	4	-	5	-	-	-	-	6	-	7	-	-	-	8	-
	Timer E	-	1	-	-	2	-	-	-	3	-	-	-	6	-	7	8	-	-	-	-	-	-	4	5
	Timer F	-	-	1	-	-	2	-	-	-	-	3	-	-	4	5	-	6	-	7	8	-	-	-	-

Figure 376 and *Figure 377* give an example of external event blanking for all edge and level sensitivities, in regular and postponed modes.

Figure 376. External trigger blanking with edge-sensitive trigger

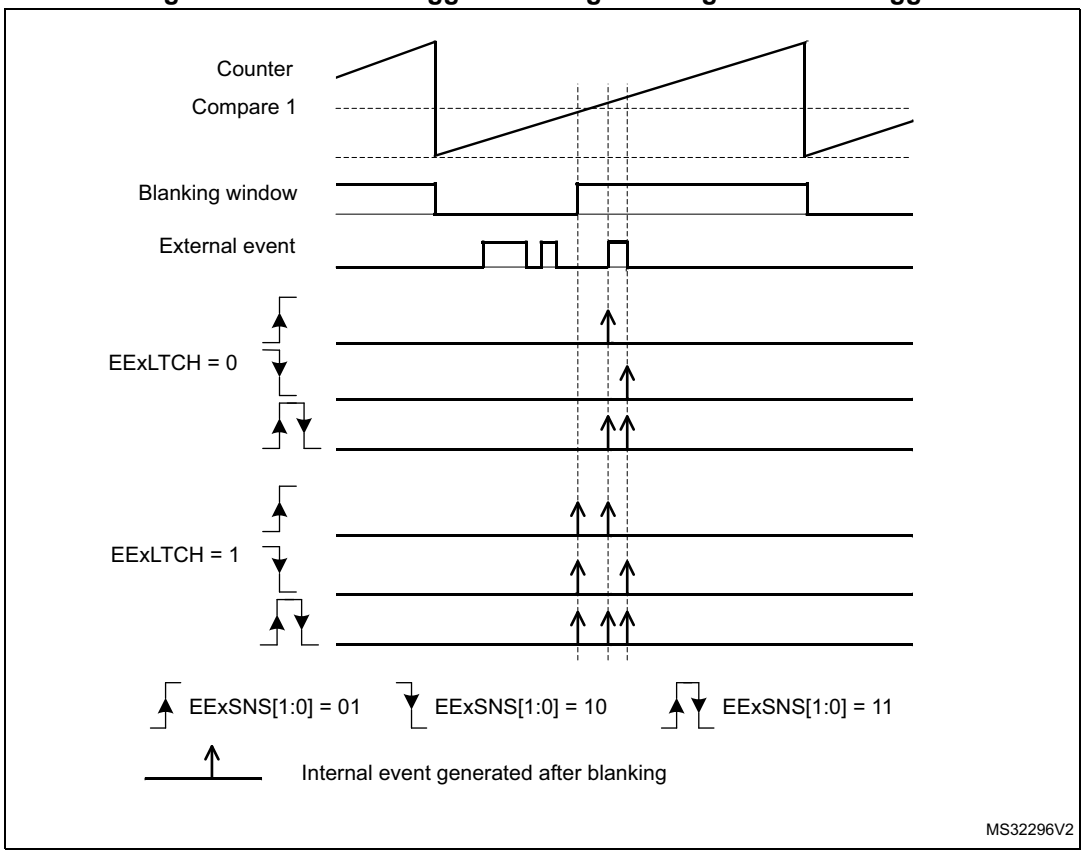
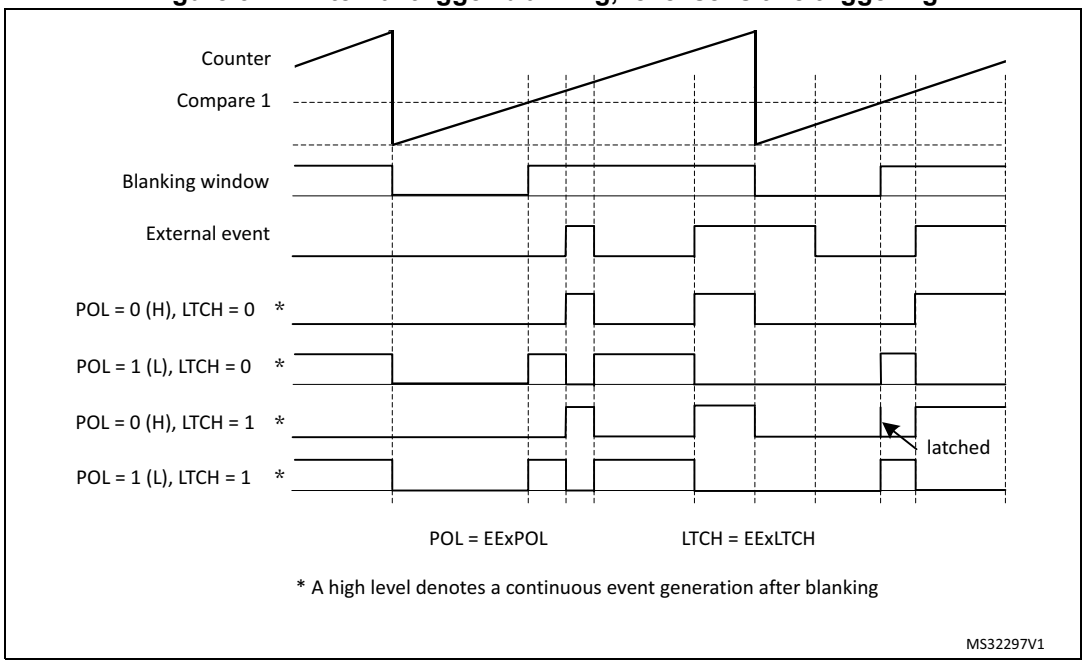


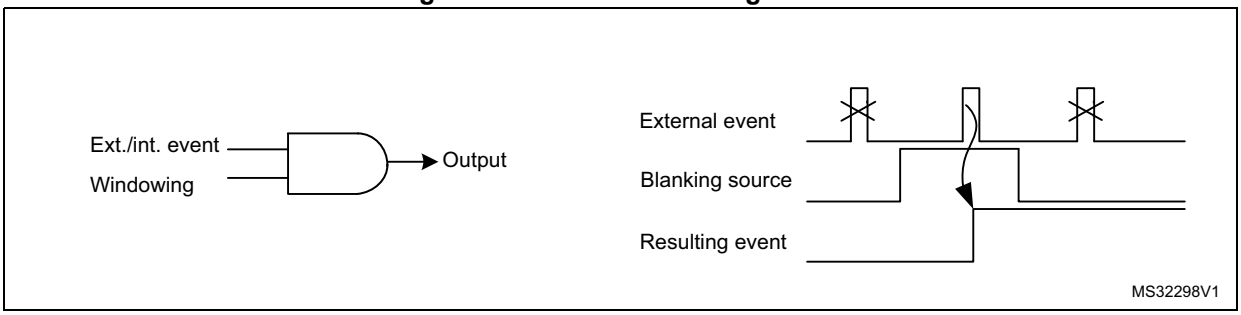
Figure 377. External trigger blanking, level sensitive triggering



Windowing mode

In event windowing mode, the event is taken into account only if it occurs within a given time window, otherwise it is ignored. This mode is active for EExFLTR[3:0] ranging from 1101 to 1111.

Figure 378. Event windowing mode



EExLTCH bit in EEFxR1 and EEFxR2 registers allows to latch the signal, if set to 1: in this case, an event is accepted if it occurs during the window but is delayed at the end of it.

- If EExLTCH bit is reset and the signal occurs during the window, it is passed through directly.
- If EExLTCH bit is reset and no signal occurs, a timeout event is generated at the end of the window.

A use case of the windowing mode is to filter synchronization signals. The timeout generation allows to force a default synchronization event, when the expected synchronization event is lacking (for instance during a converter start-up).

There are 3 sources for each external event windowing, coded as follows:

- 1101 and 1110: the windowing lasts from the counter reset to the compare match (respectively compare 2 and compare 3). In up/down mode (UDM bit set to 1), the counter reset event is defined as per the ROM[1:0] bit setting.
- 1111: the windowing is related to another timing unit and lasts from its counter reset to its compare 2 match. The source is described as TIMWIN in the bit description and is given in *Table 394*. As an example, the external events in timer B can be filtered by a window starting from timer A counter reset to timer A compare 2.

Table 394. Windowing signals mapping per timer (EEFLTR[3:0] = 1111)

Destination	Timer A	Timer B	Timer C	Timer D	Timer E	Timer F
TIMWIN (source)	Timer B CMP2	Timer A CMP2	Timer D CMP2	Timer C CMP2	Timer F CMP2	Timer E CMP2

Note: The timeout event generation is not supported if the external event is programmed in fast mode.

Figure 379 and *Figure 380* present how the events are generated for the various edge and level sensitivities, as well as depending on EExLTCH bit setting. Timeout events are specifically mentioned for clarity reasons.

Figure 379. External trigger windowing with edge-sensitive trigger

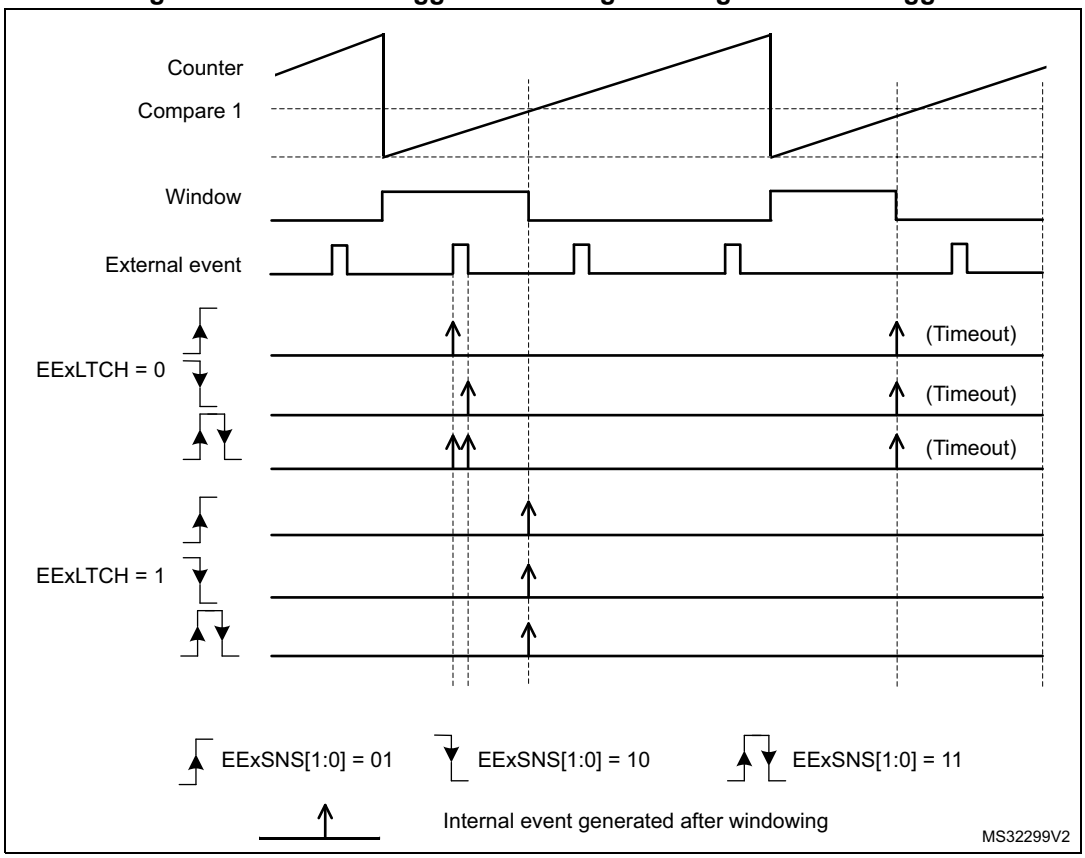
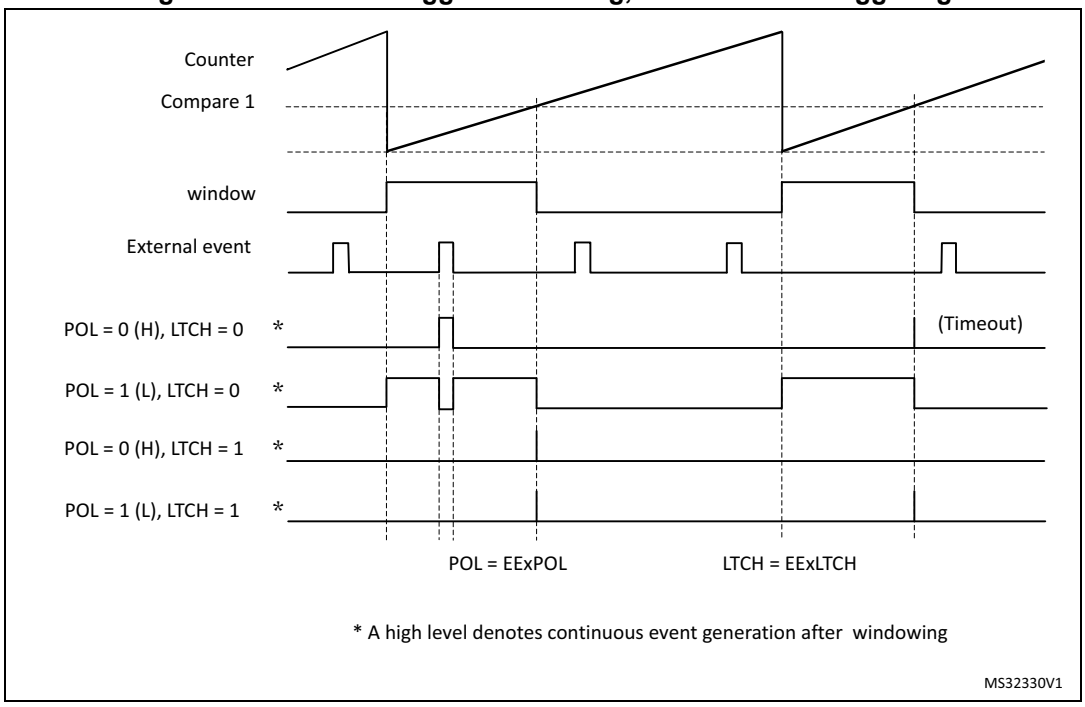


Figure 380. External trigger windowing, level sensitive triggering

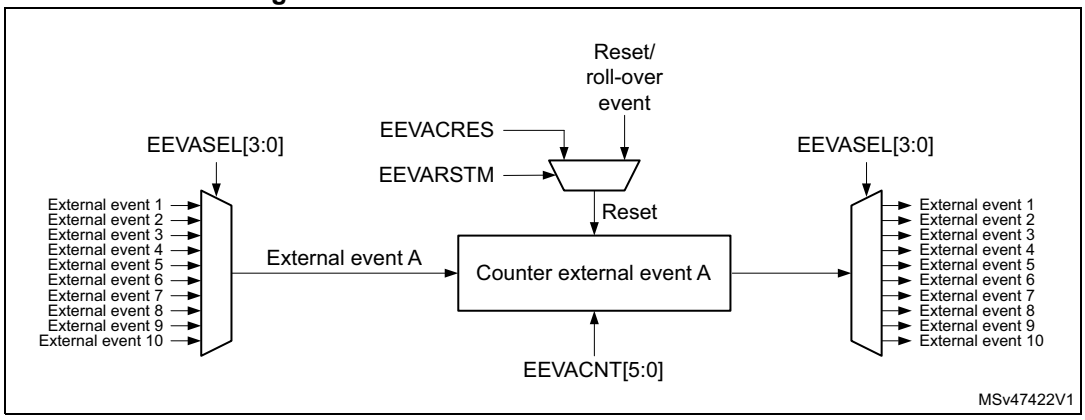


External event counter

Each timing unit also features an external event counter following the filtering unit, typically for valley skipping implementation.

The circuitry allows to filter any of the 10 external events filtered, as shown in *Figure 381*.

Figure 381. External event counter – channel A



The counter is enabled using the EEVACE bit in the HRTIM_EEFxR3 register. This mode is only valid for edge-sensitive external events (EEASNS[1:0] bit = 01,10 or 11).

The external event is propagated to the timer only if the number of active edges is greater than or equal to the value programmed in (EEVACNT[5:0]+1).

Two operating modes are available:

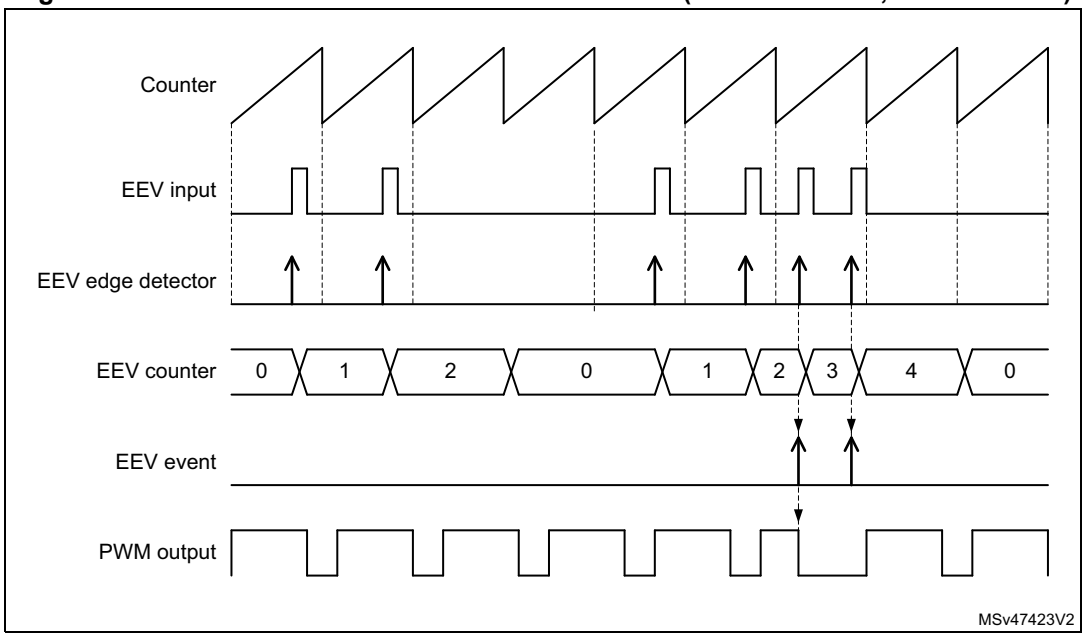
- When the EEVARSTM bit is reset, the external event counter is reset on each reset/roll-over event: the external event is active only if it appears several times within a given PWM period
- When the EEVARSTM/ bit is set, the external event counter is reset only if the event did not appear during the last PWM period. This a cumulative mode, where the event must occur at least once during multiple PWM period, as shown in *Figure 382* below.

The external event counter must be enabled after having programmed the counter value (the EEVACE bit must be set after having written the EEVACNT[5:0] bits).

Once the counter is enabled, the EEVACNT[5:0] bits can then be changed on-the-fly at any time. The new value is taken into account on the following reset/rollover event as per the EEVARSTM bit programming, or after a software reset (EEVACRES bit set).

The EEVASEL[3:0]bits must not be modified once the EEVACE bit is set.

Figure 382. External event counter cumulative mode (EEVxRSTM = 1, EEVxCNT = 2)



34.3.10 Delayed protection

The HRTIM features specific protection schemes, typically for resonant converters when it is necessary to shut down the PWM outputs in a delayed manner, either once the active pulse is completed or once a push-pull period is completed. These features are enabled with DLYPRTEN bit in the HRTIM_OUTxR register, and are using specific external event channels.

Delayed idle

In this mode, the active pulse is completed before the protection is activated. The selected external event causes the output to enter in idle mode at the end of the active pulse (defined by an output reset event in HRTIM_RSTx1R or HRTIM_RSTx2R).

Once the protection is triggered, the idle mode is permanently maintained but the counter continues to run, until the output is re-enabled. Tx1OEN and Tx2OEN bits are not affected by the delayed idle entry. To exit from delayed idle and resume operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1. The output state changes on the first transition to an active state following the output enable command.

Note: The delayed idle mode cannot be exited immediately after having been entered, before the active pulse is completed: it is mandatory to make sure that the outputs are in idle state before resuming the run mode. This can be done by waiting up to the next period, for instance, or by polling the O1CPY and/or O2CPY status bits in the TIMxISR register.

The delayed idle mode can be applied to a single output (DLYPRT[2:0] = x00 or x01) or to both outputs (DLYPRT[2:0] = x10).

An interrupt or a DMA request can be generated in response to a Delayed Idle mode entry. The DLYPRT flag in HRTIM_TIMxISR is set as soon as the external event arrives, independently from the end of the active pulse on output.

When the Delayed Idle mode is triggered, the output states can be determined using O1STAT and O2STAT in HRTIM_TIMxISR. Both status bits are updated even if the delayed

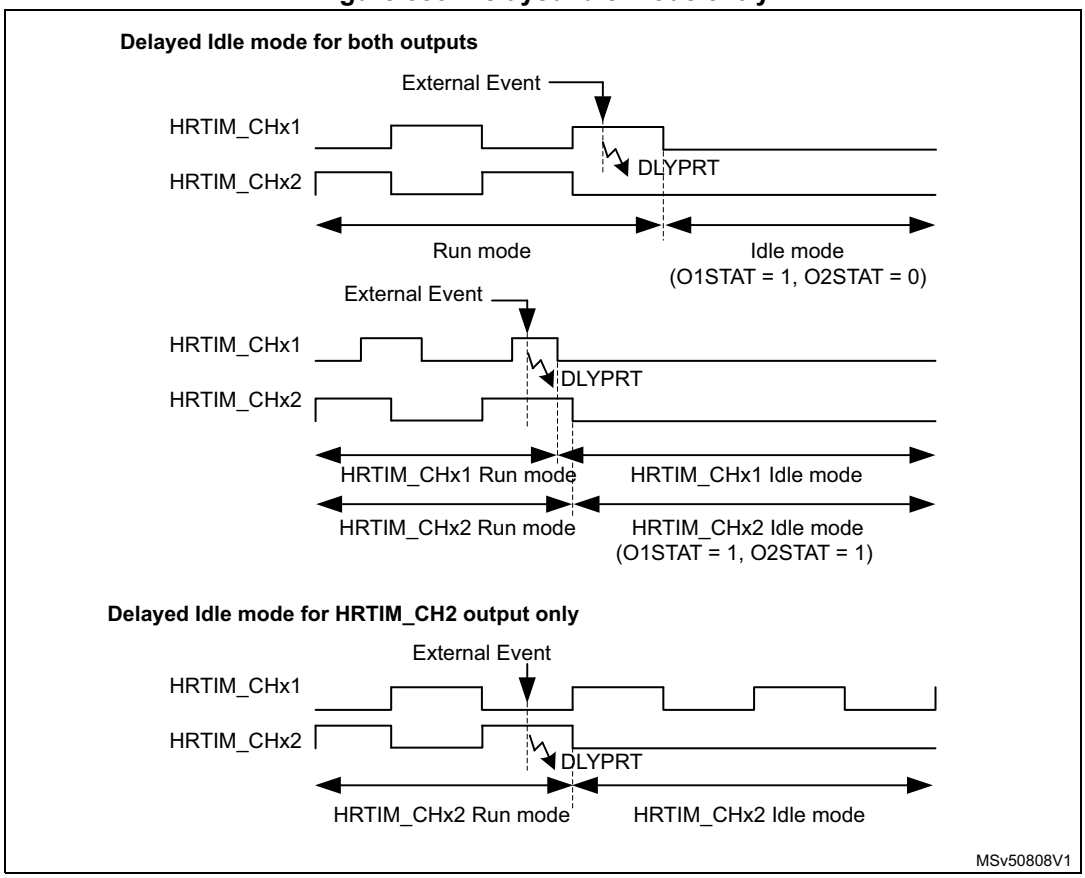
idle is applied to a single output. When the push-pull mode is enabled, the IPPSTAT flag in HRTIM_TIMxISR indicates during which period the delayed protection request occurred.

This mode is available whatever the timer operating mode (regular, push-pull, deadtime). It is available with 2 external events only:

- hrtim_eev6 and hrtim_eev7 for timer A, B and C
- hrtim_eev8 and hrtim_eev9 for timer D, E and F

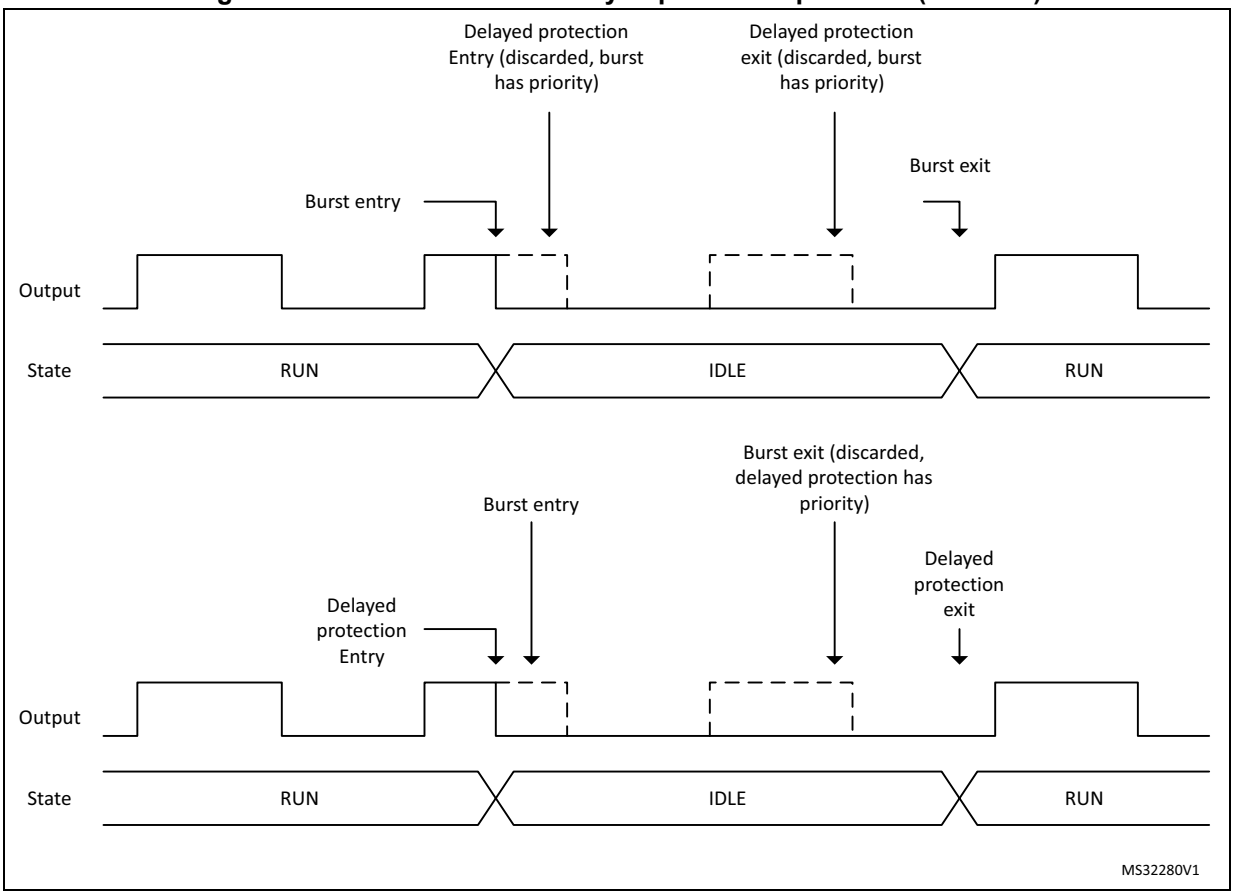
The delayed protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxEN bit is reset, until the TxyOEN bits are set.

Figure 383. Delayed Idle mode entry



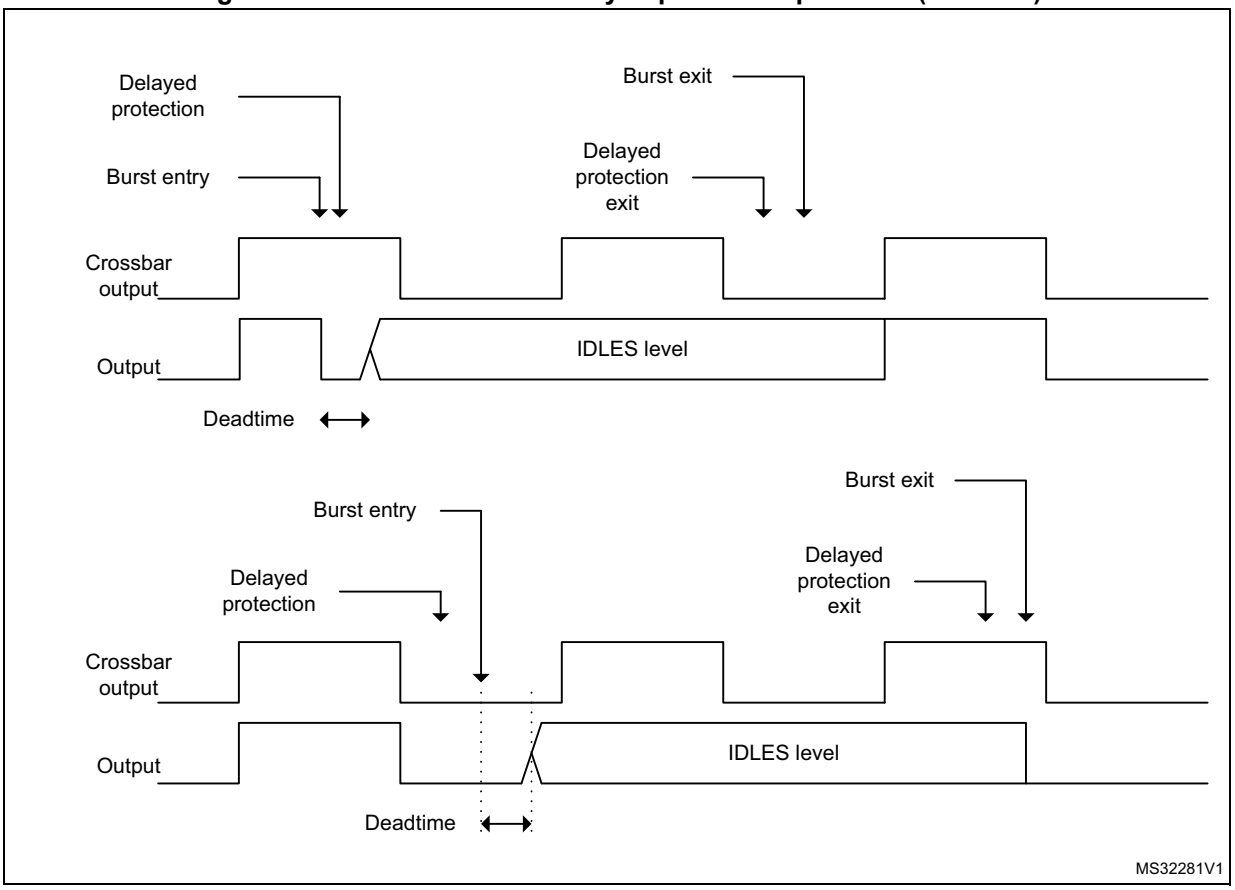
The delayed idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the delayed idle protection has been triggered. On the contrary, If the delayed protection is exited while the burst mode is active, the burst mode is resumed normally and the output is maintained in the idle state until the burst mode exits. *Figure 384* gives an overview of these different scenarios.

Figure 384. Burst mode and delayed protection priorities (DIDL = 0)



The same priorities are applied when the delayed burst mode entry is enabled (DIDL bit set), as shown in *Figure 385* below.

Figure 385. Burst mode and delayed protection priorities (DIDL = 1)



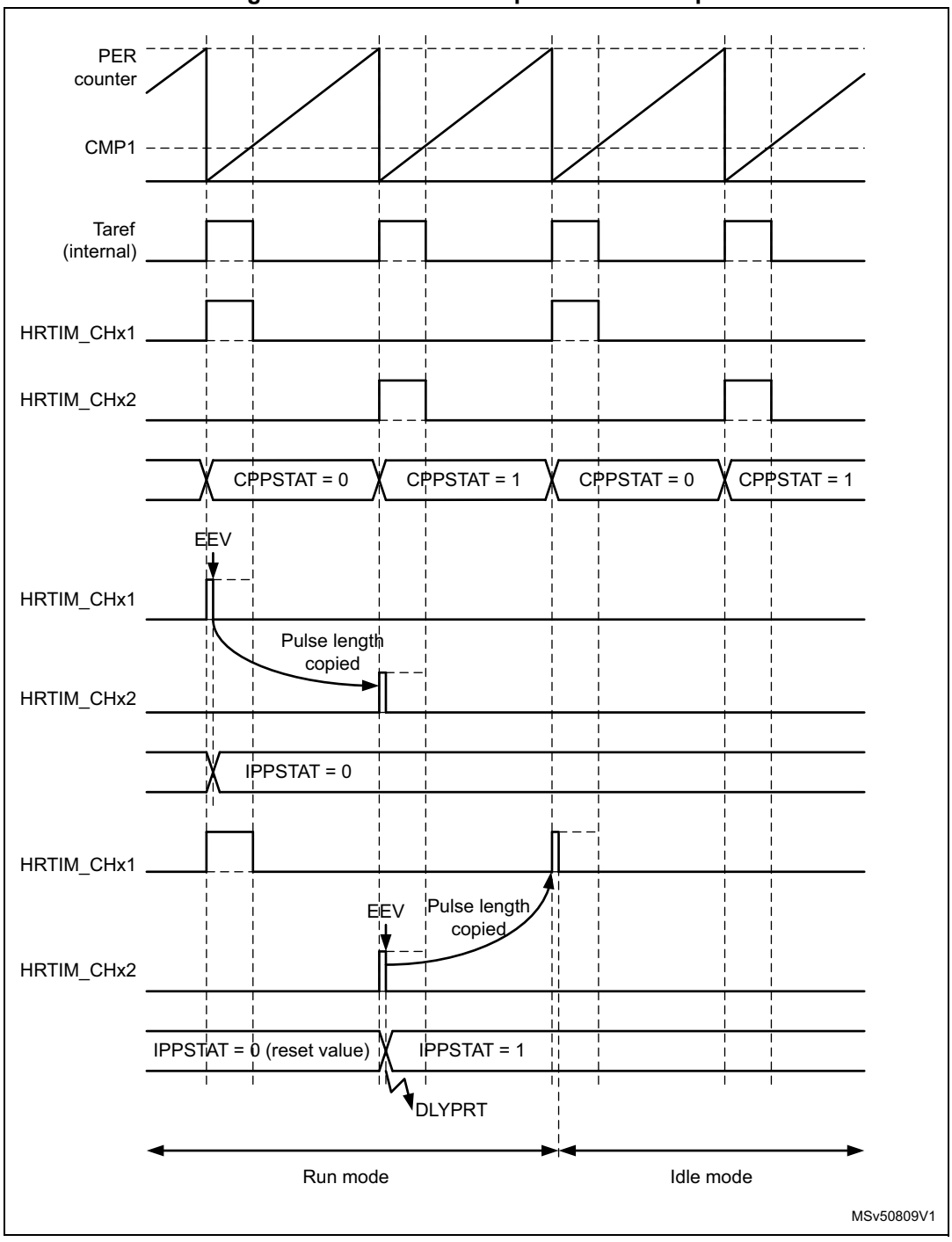
Balanced idle

Only available in push-pull mode, the balanced idle allows to have a balanced pulsewidth on the two outputs when one of the active pulse is shortened due to a protection. The pulsewidth, terminated earlier than programmed, is copied on the alternate output, then the two outputs are put in idle state, until the normal operation is resumed by software. This mode is enabled by writing x11 in DLYPRT[2:0] bit field in HRTIM_OUTxR.

This mode is available with only 2 external events:

- hrtim_eev6 and hrtim_eev7 for timer A, B and C
- hrtim_eev8 and hrtim_eev9 for timer D, E and F

Figure 386. Balanced Idle protection example



When the balanced Idle mode is enabled, the selected external event triggers a capture of the counter value into the compare 4 active registers (this value is not user-accessible). The push-pull is maintained for one additional period so that the shorten pulse can be repeated: a new output reset event is generated while the regular output set event is maintained.

The Idle mode is then entered and the output takes the level defined by IDLESx bits in the HRTIM_OUTxR register. The balanced idle mode entry is indicated by the DLYPRT flag,

while the IPPSTAT flag indicates during which period the external event occurred, to determine the sequence of shorten pulses (A1 then A2 or vice versa).

The timer operation is not interrupted (the counter continues to run).

To enable the balanced idle mode, it is necessary to have the following initialization:

- Timer operating in continuous mode (CONT = 1)
- Push-pull mode enabled
- HRTIM_CMP4xR must be set to 0 and the content transferred into the active register (for instance by forcing a software update)
- DELCMP4[1:0] bit field must be set to 00 (auto-delayed mode disabled)
- DLYPRT[2:0] = x11 (delayed protection enable)

Note: The HRTIM_CMP4xR register must not be written during a balanced idle operation. The CMP4 event is reserved and cannot be used for another purpose.

In balanced idle mode, it is recommended to avoid multiple external events or software-based reset events causing an output reset. If such an event arrives before a balanced idle request within the same period, it causes the output pulses to be unbalanced (1st pulse length defined by the external event or software reset, while the 2nd pulse is defined by the balanced idle mode entry).

The minimum pulsewidth that can be handled in balanced idle mode is 4 f_{HRTIM} clock periods (0x80 when CKPSC[2:0] = 0, 0x40 if CKPSC[2:0] = 1, 0x20 if CKPSC[2:0] = 2,...).

If the capture occurs before the counter has reached this minimum value, the current pulse is extended up to 4 f_{HRTIM} clock periods before being copied into the secondary output. In any case, the pulse widths are always balanced.

Tx1OEN and Tx2OEN bits are not affected by the balanced idle entry. To exit from balanced idle and resume the operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1 simultaneously. The output state changes on the first active transition following the output enable.

It is possible to resume operation similarly to the delayed idle entry. For instance, if the external event arrives while output 1 is active (delayed idle effective after output 2 pulse), the re-start sequence can be initiated for output 1 first. To do so, it is necessary to poll CPPSTAT bit in the HRTIM_TIMxISR register. Using the above example (IPPSTAT flag equal to 0), the operation is resumed when CPPSTAT bit is 0.

In order to have a specific re-start sequence, it is possible to poll the CPPSTAT to know which output is active first. This allows, for instance, to re-start with the same sequence as the idle entry sequence: if the external event arrives during output 1 active, the re-start sequence is initiated when the output 1 is active (CPPSTAT = 0).

Note: The balanced idle mode must not be disabled while a pulse balancing sequence is on-going. It is necessary to wait until the CMP4 flag is set, thus indicating that the sequence is completed, to reset the DLYPRTEN bit.

The balanced idle protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxCEN bit is reset, until TxyOEN bits are set.

Balanced idle can be used together with the burst mode under the following conditions:

- TxBM bit must be reset (counter clock maintained during the burst, refer to *Section 34.3.15*),
- No balanced idle protection must be triggered while the outputs are in a burst idle state.

The balanced idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the balanced idle protection has been triggered. On the contrary, if the delayed protection is exited while the burst mode is active, the burst mode is resumed normally.

Note: Although the output state is frozen in idle mode, a number of events are still generated on the auxiliary outputs (refer to Section 34.3.18) during the idle period following the delayed protection:
- Output set/reset interrupt or DMA requests
- External event filtering based on output signal
- Capture events triggered by set/reset

Balanced idle automatic resuming

The balanced Idle mode can be configured to have an automatic resuming of operation after a trigger.

Once the shorten pulse has been copied to the alternate output, the pulse width is reset to its original value and the timer resumes operation: the two outputs keep on being in RUN mode.

This is enabled by setting the BIAR bit in the HRTIM_OUTxR register.

This mode must be used only when the period in HRTIM_PERxR is greater than 6 periods of the fHRTIM clock, that is 0xC0 if CKPSC[2:0] = 0, 0x60 if CKPSC[2:0] = 1, 0x30 if CKPSC[2:0] = 2,...

Note: This bit is only significant if DLYPRT[2:0] = 011 or 111, it is ignored otherwise.

In balanced idle automatic resuming mode, it is mandatory to set the IDLES state to inactive.

34.3.11 Register preload and update management

Most of HRTIM registers are buffered and can be preloaded if needed. Typically, this allows to prevent the waveforms from being altered by a register update not synchronized with the active events (set/reset).

When the preload mode is enabled, accessed registers are shadow registers. Their content is transferred into the active register after an update request, either software or synchronized with an event.

By default, PREEN bits in HRTIM_MCR and HRTIM_TIMxCR registers are reset and the registers are not preloaded: any write directly updates the active registers. If PREEN bit is reset while the timer is running and preload was enabled, the content of the preload registers is directly transferred into the active registers.

Each timing unit and the master timer have their own PREEN bit. If PREEN is set, the preload registers are enabled and transferred to the active register only upon an update event.

There are two options to initialize the timer when the preload feature is needed:

- Enable PREEN bit at the very end of the timer initialization to have the preload registers transferred into the active registers before the timer is enabled (by setting MCEN and TxCEN bits).
- enable PREEN bit at any time during the initialization and force a software update immediately before starting.

Table 395 lists the registers which can be preloaded, together with a summary of available update events.

Table 395. HRTIM preloadable control registers and associated update sources

Timer	Preloadable registers	Preload enable	Update sources
Master timer	HRTIM_MDIER HRTIM_MPER HRTIM_MREP HRTIM_MCMP1R HRTIM_MCMP2R HRTIM_MCMP3R HRTIM_MCMP4R	PREEN bit in HRTIM_MCR	Software Repetition event Burst DMA event Repetition event following a burst DMA event
Timer x x = A..F	HRTIM_TIMxDIER HTRIM_PERxR HTRIM_REPxR HTRIM_CMP1xR HTRIM_CMP1CxR HTRIM_CMP2xR HTRIM_CMP3xR HTRIM_CMP4xR HRTIM_DTxR HRTIM_SETx1R HRTIM_RSTx1R HRTIM_SETx2R HRTIM_RSTx2R HRTIM_RSTxR	PREEN bit in HRTIM_TIMxCR	Software TIMx repetition event TIMx reset event Burst DMA event Update event from other timers (TIMy, master) Update event following a burst DMA event Update enable inputs hrtim_upd_en[3:1] Update event following an update enable input following an update event on hrtim_upd_en[3:1] inputs
HRTIM Common	HRTIM_ADC1R HRTIM_ADC2R HRTIM_ADC3R HRTIM_ADC4R	TIMx or master timer Update, depending on ADxUSRC[2:0] bits in HRTIM_CR1, if PREEN = 1 in the selected timer	

The master timer has four update options:

1. Software: writing 1 into MSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is cancelled.
2. Update done when the master counter rolls over and the master repetition counter is equal to 0. This is enabled when MREPU bit is set in HRTIM_MCR.
3. Update done once burst DMA is completed (refer to *Section 34.3.23* for details). This is enabled when BRSTDMA[1:0] = 01 in HRTIM_MCR. It is possible to have both MREPU=1 and BRSTDMA=01.

Note: The update can take place immediately after the end of the burst sequence if SWU bit is set (that is forced update mode). If SWU bit is reset, the update is done on the next update event following the end of the burst sequence.

4. Update done when the master counter rolls over following a burst DMA completion. This is enabled when BRSTDMA[1:0] = 10 in HRTIM_MCR.

An interrupt or a DMA request can be generated by the master update event.

Each timer (TIMA..F) can also have the update done as follows:

- By software: writing 1 into TxSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is canceled.
- Update done when the counter rolls over and the repetition counter is equal to 0. This is enabled when TxREPU bit is set in HRTIM_TIMxCR.
- Update done when the counter is reset or rolls over in continuous mode. This is enabled when TxRSTU bit is set in HRTIM_TIMxCR. This is used for a timer operating in single-shot mode, for instance.
- Update done once a burst DMA is completed. This is enabled when UPDGAT[3:0] = 0001 in HRTIM_TIMxCR.
- Update done on the update event following a burst DMA completion (the event can be enabled with TxRSTU, TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0010 in HRTIM_TIMxCR.
- Update done when receiving a request on hrtim_upd_en[3:1]. This is enabled when UPDGAT[3:0] = 0011, 0100, 0101 in HRTIM_TIMxCR.
- Update done on the update event following a request on hrtim_upd_en[3:1] (the event can be enabled with TxRSTU, TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0110, 0111, 1000 in HRTIM_TIMxCR
- Update done synchronously with any other timer or master update (for instance TIMA can be updated simultaneously with TIMB). This is used for converters requiring several timers, and is enabled by setting bits MSTU and TxU in HRTIM_TIMxCR register.

The update enable inputs hrtim_upd_en[3:1] allow to have an update event synchronized with on-chip events coming from the general-purpose timers. These inputs are rising-edge sensitive.

Table 380 lists the connections between update enable inputs and the on-chip sources.

This allows to synchronize low frequency update requests with high-frequency signals (for instance an update on the counter roll-over of a 100 kHz PWM that has to be done at a 100 Hz rate).

Note: The update events are synchronized to the prescaler clock when CKPSC[2:0] > 5.

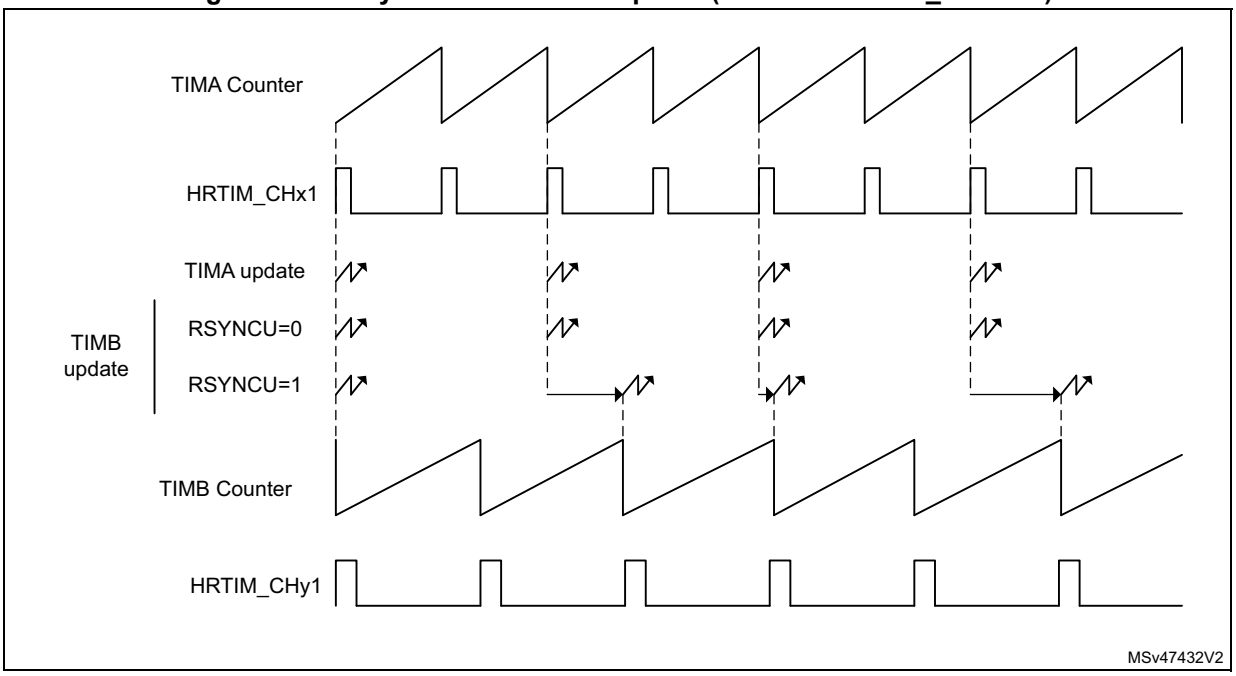
The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) can either be taken into account immediately or re-synchronized with the timers reset/roll-over event. This is done with the RSYNCU bit in the HRTIM_TIMxCR register, as show in *Figure 387* below):

- RSYNCU = 0: the update coming from adjacent timers is taken into account immediately
- RSYNCU = 1: the update coming from adjacent timers is taken into account on the following reset/roll-over event.

The RSYNCU bit is significant only when UPDGAT[3:0] = 0000, it is ignored otherwise.

An interrupt or a DMA request can be generated by the Timx update event.

Figure 387. Resynchronized timer update (TAU=1 in HRTIM_TIMBCR)



MUDIS and TxUDIS bits in the HRTIM_CR1 register allow to temporarily disable the transfer from preload to active registers, whatever the selected update event is. This allows to modify several registers in multiple timers. The regular update event takes place once these bits are reset.

MUDIS and TxUDIS bits are all grouped in the same register. This allows the update of multiple timers (not necessarily synchronized) to be disabled and resumed simultaneously.

The following example is a practical use case. A first power converter is controlled with the master, TIMB and TIMC. TIMB and TIMC must be updated simultaneously with the master timer repetition event. A second converter works in parallel with TIMA, TIMD and TIME, and TIMD, TIME must be updated with TIMA repetition event.

First converter:

In HRTIM_MCR, MREPU bit is set: the update occurs at the end of the master timer counter repetition period. In HRTIM_TIMBCR and HRTIM_TIMCCR, MSTU bits are set to have TIMB and TIMC timers updated simultaneously with the master timer.

When the power converter set-point has to be adjusted by software, MUDIS, TBUDIS and TCUDIS bits of the HRTIM_CR register must be set prior to write accessing registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, MUDIS, TBUDIS and TCUDIS bits must be reset. The transfer from preload to active registers is done as soon as the master repetition event occurs.

Second converter:

In HRTIM_TIMACR, TAREPU bit is set: the update occurs at the end of the timer A counter repetition period. In HRTIM_TIMDCR and HRTIM_TIMECR, TAU bits are set to have TIMD and TIME timers updated simultaneously with timer A.

When the power converter set-point has to be adjusted by software, TAUDIS, TDUDIS and TEUDIS bits of the HRTIM_CR register must be set prior to write accessing the registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, TAUDIS, TDUDIS and TEUDIS bits can be reset: the transfer from preload to active registers is done as soon as the timer A repetition event occurs.

34.3.12 PWM mode with “greater than” comparison

A specific no-latency update mode is available for PWM signals generated with the CMP1 and CMP3 registers. It allows to have a new duty cycle value applied as soon as possible within the PWM cycle, without having to wait the completion of the current PWM period. This reduces the overall delay time in software control loops. As shown in *Figure 388* below, this eventually allows to have:

- An early turn-off of the output if the new compare value is below the current counter value and the current compare value is above the counter, at the time the new value is written.
- An early turn-on of the output, re-enabling the output if the new compare value is above the counter value and the current compare value is above the counter, at the time the new value is written.

The output signal is left unchanged when the new compare value and current compare value are both below the counter.

This feature is only available for CMP1 or CMP3 RESET events, and is enabled using the GTCMP1 and GTCMP3 enable bits in the HRTIM_TIMxCR2 register.

The preload mechanism is inactive for a compare register when the corresponding GTCMPx bit is set, whatever the PREEN bit value. This mode is intended to have the new compare value taken into account as soon as possible after a new value write, without waiting for the preload to active register transfer.

These bits are defining the compare 1 and compare 3 operating modes as follows:

- GTCMPx = 0: the compare x event is generated when the counter is equal to the compare value (compare match mode). If the compare value is changed on-the-fly, the compare event may not be generated.
- GTCMPx = 1: the compare x event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

The “greater than” compare mode causes the crossbar to act differently depending on the comparison result. Let’s consider the CMP1 event is doing an output RESET. When the new compare value is written, two cases are considered

- If the new compare value is below the counter value, the RESET event is issued and can eventually cause an early turn-OFF
- If the new compare value is above the counter value, a SET event is generated so as to re-arm the output value before it is actually RESET when the counter exceeds the counter value (early turn-ON).

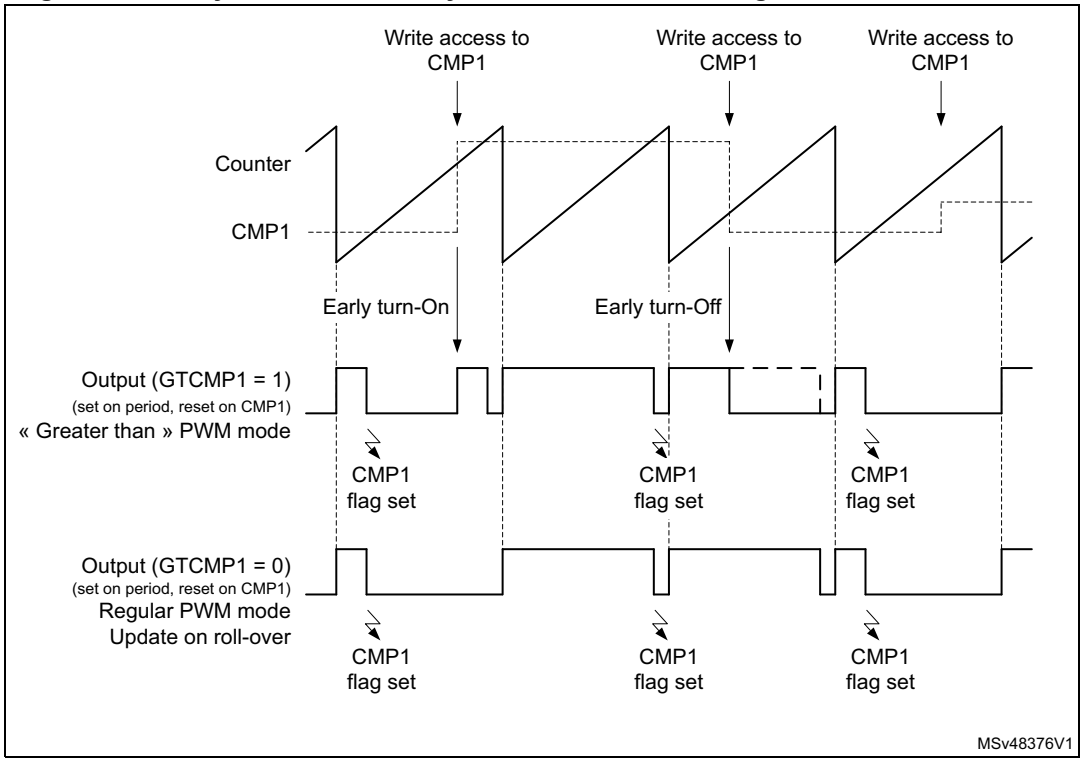
The “greater than” compare mode is supported for both SET and RESET actions.

The “greater than” compare mode must only be used for the following configurations:

1. In the fixed frequency configuration, the period event must trigger the output set and the “greater than” compare triggers the output reset (or vice versa the period must trigger the reset if the “greater than” compare triggers the set).
2. For variable frequency configuration, the event selected as counter reset source must also be selected as set or reset source for the timer output (opposite direction as the “greater than” compare event).

Note: The “greater than” modes must not be used when the CMP1 and/or CMP3 modes are controlled by hardware in half and interleaved modes.

Figure 388. Early turn-ON and early turn-OFF behavior in “greater than” PWM mode



The immediate update mode implies that the content of the preload register is transferred into the active register at the very same time the register is written. When GTCMP1 and/or GTCMP3 bits are set, their respective preload mechanism is disabled (for HRTIM_TIMxCMP1 and/or HRTIM_TIMxCMP3 registers), whatever the PREEN bit value is.

Note: The compare interrupt flags (CMP1 and CMP3 in HRTIM_TIMxISR) are not generated in case of late turn-ON and early turn-OFF, as shown in Figure 388.

The “Greater than” comparison must not be done on both CMP1 and CMP3 for the same output (GTCMP1 and GTCMP3 bits must not be set simultaneously).

34.3.13 Events propagation within or across multiple timers

The HRTIM offers many possibilities for cascading events or sharing them across multiple timing units, including the master timer, to get full benefits from its modular architecture. These are key features for converters requiring multiple synchronized outputs.

This section summarizes the various options and specifies whether and how an event is propagated within the HRTIM.

TIMx update triggered by the master timer update

The sources listed in *Table 396* are generating a master timer update. The table indicates if the source event can be used to trigger a simultaneous update in any of TIMx timing units.

Operating condition: MSTU bit is set in HRTIM_TIMxCR register.

Table 396. Master timer update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	BRSTDMA[1:0] = 01	No	Must be done in TIMxCR (UPDGAT[3:0] = 0001)
Roll-over event following a burst DMA end	BRSTDMA[1:0] = 10	Yes	-
Repetition event caused by a counter roll-over	MREPU = 1	Yes	-
Repetition event caused by a counter reset (from HRTIM_SCIN or software)		No	-
Software update	MSWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx update triggered by the TIMy update

The sources listed in *Table 397* are generating a TIMy update. The table indicates if the given event can be used to trigger a simultaneous update in another or multiple TIMx timers.

Operating condition: TyU bit set in HRTIM_TIMxCR register (source = TIMy and destination = TIMx).

Table 397. TIMx update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0001)
Update caused by the update enable inputs hrtim_upd_en[3:1]	UPDGAT[3:0] = 0011, 0100, 0101	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0011, 0100, 0101
Master update	MSTU = 1 in HRTIM_TIMyCR	No	Must be done with MSTU = 1 in HRTIM_TIMxCR
Another TIMx update (TIMz>TIMy>TIMx)	TzU=1 in HRTIM_TIMyCR TyU=1 in TIMxCR	No	Must be done with TzU=1 in HRTIM_TIMxCR TzU=1 in HRTIM_TIMyCR
Repetition event caused by a counter roll-over	TyREPU = 1	Yes	-

Table 397. TIMx update event propagation (continued)

Source	Condition	Propagation	Comment
Repetition event caused by a counter reset	TyREPU = 1	-	Refer to counter reset cases below
Counter roll-over	TyRSTU = 1	Yes	-
Counter software reset	TyRST=1 in HRTIM_CR2	No	Can be done simultaneously with update in HRTIM_CR2 register
Counter reset caused by a TIMz compare	TIMzCMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by external events	EXTEVNTn in HRTIM_RSTyR	Yes	-
Counter reset caused by a master compare or a master period	MSTCMPn or MSTPER in HRTIM_RSTyR	Yes	-
Counter reset caused by a TIMy compare	CMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by an update	UPDT in HRTIM_RSTyR	No	Propagation would result in a lock-up situation (update causing reset causing update)
Counter reset caused by HRTIM_SCIN	SYNCRSTy in HRTIM_TIMyCR	No	-
Software update	TySWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx counter reset causing a TIMx update

Table 398 lists the counter reset sources and indicates whether they can be used to generate an update.

Operating condition: TxRSTU bit in HRTIM_TIMxCR register.

Table 398. Reset events able to generate an update

Source	Condition	Propagation	Comment
Counter roll-over	-	Yes	-
Update event	UPDT in HRTIM_RSTxR	No	Propagation would result in a lock-up situation (update causing a reset causing an update)
External event	EXTEVNTn in HRTIM_RSTxR	Yes	-
TIMy compare	TIMyCMPn in HRTIM_RSTxR	Yes	-
Master compare	MSTCMPn in HRTIM_RSTxR	Yes	-
Master period	MSTPER in HRTIM_RSTxR	Yes	-
Compare 2 and 4	CMPn in HRTIM_RSTxR	Yes	-
Software	TxRST=1 in HRTIM_CR2	Yes	-
HRTIM_SCIN	SYNCRSTx in HRTIM_TIMxCR	Yes	-

TIMx update causing a TIMx counter reset

Table 399 lists the update event sources and indicates whether they can be used to generate a counter reset.

Operating condition: UPDT bit set in HRTIM_RSTxR.

Table 399. Update event propagation for a timer reset

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	Yes	-
Update caused by the update enable inputs hrtim_upd_en[3:1]	UPDGAT[3:0] = 0011, 0100, 0101	Yes	-
Master update caused by a roll-over after a burst DMA	MSTU = 1 in HRTIM_TIMxCR BRSTDMA[1:0] = 10 in HRTIM_MCR	Yes	-

Table 399. Update event propagation for a timer reset (continued)

Source	Condition	Propagation	Comment
Master update caused by a repetition event following a roll-over	MSTU = 1 in HRTIM_TIMxCR MREPU = 1 in HRTIM_MCR	Yes	-
Master update caused by a repetition event following a counter reset (software or due to HRTIM_SCIN)		No	-
Software triggered master timer update	MSTU = 1 in HRTIM_TIMxCR MSWU = 1 in HRTIM_CR2	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update
TIMy update caused by a TIMy counter roll-over	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by a TIMy repetition event	TyU = 1 in HRTIM_TIMxCR TyREPU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by an external event or a TIMy compare (through a TIMy reset)	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR EXTEVNTn or CMP4/2 in HRTIM_RSTyCR	Yes	-
TIMy update caused by sources other than those listed above	TyU = 1 in HRTIM_TIMxCR	No	-
Repetition event following a roll-over	TxREPU = 1 in HRTIM_TIMxCR	Yes	-
Repetition event following a counter reset		No	-
Timer reset	TxRSTU = 1 in HRTIM_TIMxCR	No	Propagation would result in a lock-up situation (reset causing an update causing a reset)
Software	TxSWU in HRTIM_CR2	No	-

34.3.14 Output management

Each timing unit controls a pair of outputs. The outputs have three operating states:

- RUN: this is the main operating mode, where the output can take the active or inactive level as programmed in the crossbar unit.
- IDLE: this state is the default operating state after an HRTIM reset, when the outputs are disabled by software or during a burst mode operation (where outputs are temporary disabled during a normal operating mode; refer to *Section 34.3.15* for more details). It is either permanently active or inactive.
- FAULT: this is the safety state, entered in case of a shut-down request on FAULTx inputs. It can be permanently active, inactive or Hi-Z.

The output status is indicated by TxyOEN bit in HRTIM_OENR register and TxyODS bit in HRTIM_ODSR register, as in *Table 400*.

Table 400. Output state programming, x= A..F, y = 1 or 2

TxyOEN (control/status) (set by software, cleared by hardware)	TxyODS (status)	Output operating state
1	x	RUN
0	0	IDLE
0	1	FAULT

TxyOEN bit is both a control and a status bit: it must be set by software to have the output in RUN mode. It is cleared by hardware when the output goes back in IDLE or FAULT mode. When TxyOEN bit is cleared, TxyODS bit indicates whether the output is in the IDLE or FAULT state. A third bit in the HRTIM_ODISR register allows to disable the output by software.

Figure 389. Output management overview

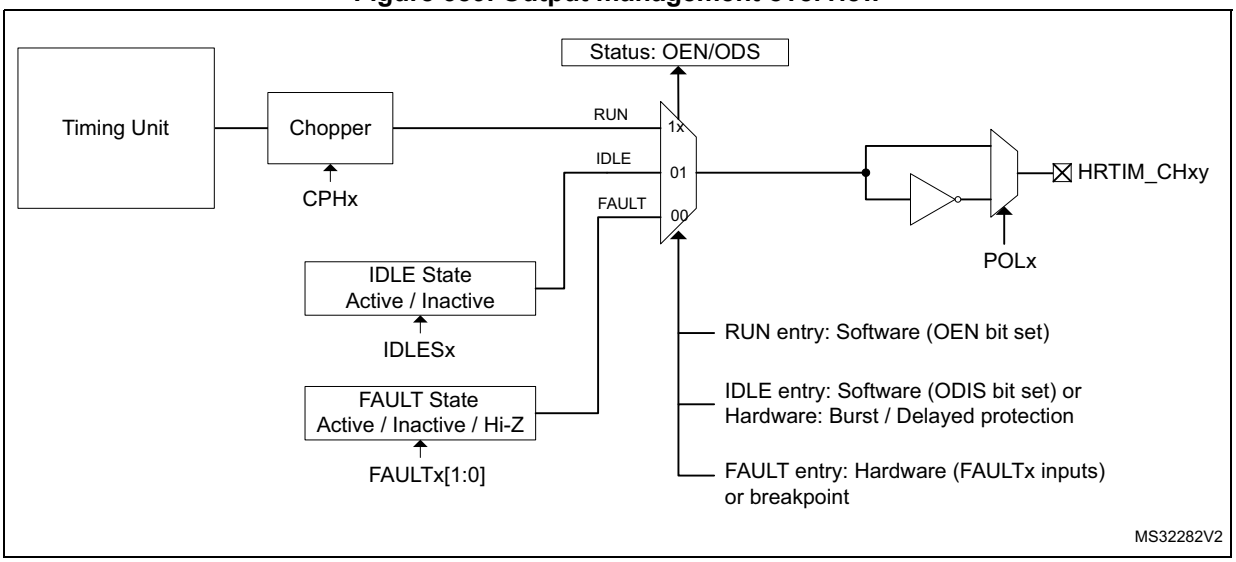
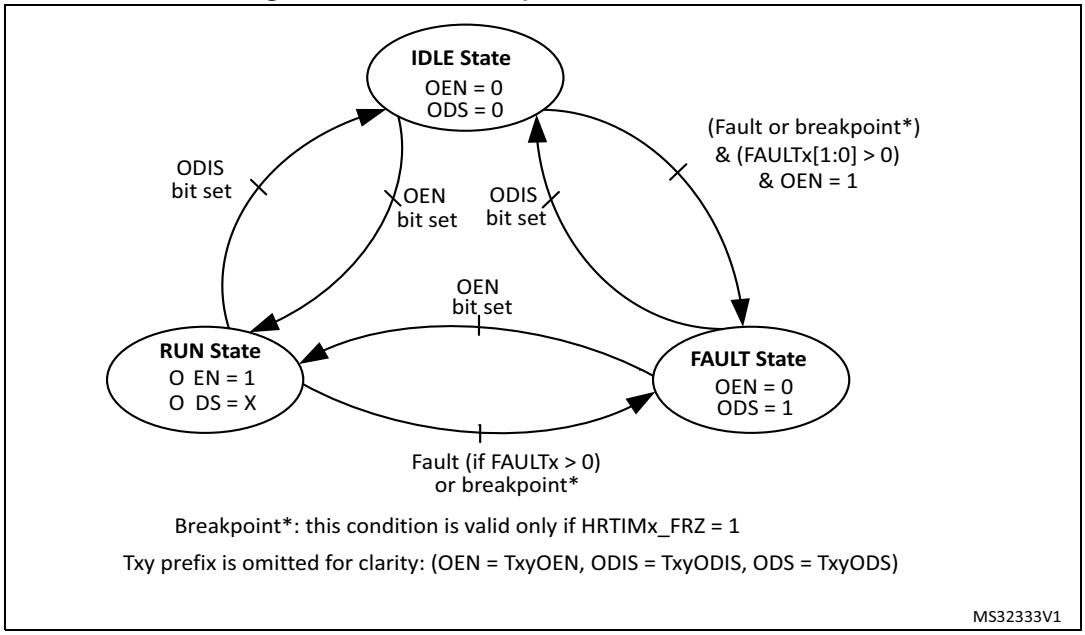


Figure 390 summarizes the bit values for the three states and how the transitions are triggered. Faults can be triggered by any external or internal fault source, as listed in

Section 34.3.17, while the Idle state can be entered when the burst mode or delayed protections are active.

Figure 390. HRTIM output states and transitions



The FAULT and IDLE levels are defined as active or inactive. Active (or inactive) refers to the level on the timer output that causes a power switch to be closed (or opened for an inactive state).

The IDLE state has the highest priority: the transition FAULT → IDLE is possible even if the FAULT condition is still valid, triggered by ODIS bit set.

The FAULT state has priority over the RUN state: if TxyOEN bit is set simultaneously with a fault event, the FAULT state is entered. The condition is given on the transition IDLE → FAULT, as in *Figure 390*: fault protection needs to be enabled (FAULTx[1:0] bits = 01, 10, 11) and the Txy OEN bit set with a fault active (or during a breakpoint if HRTIMx_FRZ = 1).

The output polarity is programmed using POLx bits in HRTIM_OUTxR. When POLx = 0, the polarity is positive (output active high), while it is active low in case of a negative polarity (POLx = 1). Practically, the polarity is defined depending on the power switch to be driven (PMOS versus NMOS) or on a gate driver polarity.

The output level in the FAULT state is configured using FAULTx[1:0] bits in HRTIM_OUTxR, for each output, as follows:

- 00: output never enters the fault state and stays in RUN or IDLE state.
- 01: output at active level when in FAULT.
- 10: output at inactive level when in FAULT.
- 11: output is tri-stated when in FAULT. The safe state must be forced externally with pull-up or pull-down resistors, for instance.

Note: FAULTx[1:0] bits must not be changed as long as the outputs are in FAULT state.

The level of the output in IDLE state is configured using IDLESx bit in HRTIM_OUTxR, as follows:

- 0: output at inactive level when in IDLE
- 1: output at active level when in IDLE

When TxyOEN bit is set to enter the RUN state, the output is immediately connected to the crossbar output. If the timer clock is stopped, the level is either inactive (after an HRTIM reset) or corresponds to the RUN level (when the timer is stopped and the output disabled).

During the HRTIM initialization, the output level can be prepositioned prior to having it in RUN mode, using the software forced output set and reset in the HRTIM_SETx1R and HRTIM_RSTx1R registers.

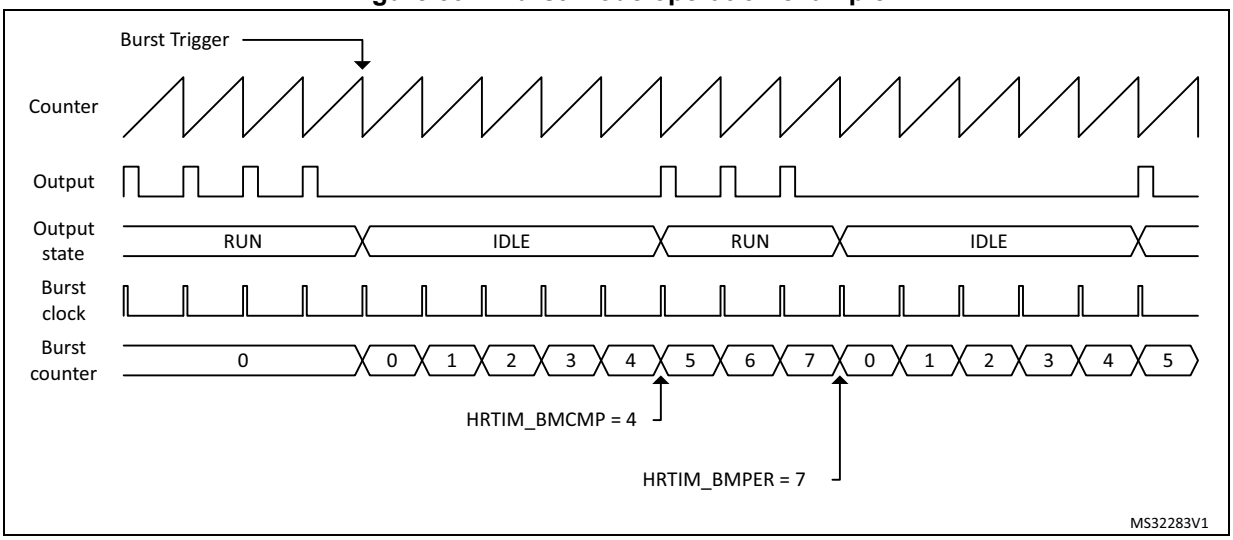
34.3.15 Burst mode controller

The burst mode controller allows to have the outputs alternatively in IDLE and RUN state, by hardware, so as to skip some switching periods with a programmable periodicity and duty cycle.

Burst mode operation is of common use in power converters when operating under light loads. It can significantly increase the efficiency of the converter by reducing the number of transitions on the outputs and the associated switching losses.

When operating in burst mode, one or a few pulses are outputs followed by an idle period equal to several counting periods, typically, where no output pulses are produced, as shown in the example on *Figure 391*.

Figure 391. Burst mode operation example



The burst mode controller consists of:

- A counter that can be clocked by various sources, either within or outside the HRTIM (typically the end of a PWM period).
- A compare register to define the number of idle periods: HRTIM_BMCMP.
- A period register to define the burst repetition rate (corresponding to the sum of the idle and run periods): HRTIM_BMPER.

The burst mode controller is able to take over the control of any of the 10 PWM outputs. The state of each output during a burst mode operation is programmed using IDLESx and IDLEMx bits in the HRTIM_OUTxR register, as in *Table 401*.

Table 401. Timer output programming for burst mode

IDLEMx	IDLESx	Output state during burst mode
0	X	No action: the output is not affected by the burst mode operation.
1	0	Output inactive during the burst
1	1	Output active during the burst

Note: IDLEMx bit must not be changed while the burst mode is active.

The burst mode controller only acts on the output stage. A number of events are still generated during the idle period:

- Output set/reset interrupt or DMA requests
- External event filtering based on Tx2 output signal
- Capture events triggered by output set/reset

During the burst mode, neither start nor reset events are generated on the hrtim_out_sync[2:1] output, even if TxBM bit is set.

Operating mode

It is necessary to have the counter enabled (TxCEN bit set) before using the burst mode on a given timing unit.The burst mode is enabled with BME bit in the HRTIM_BMCR register.

It can operate in continuous or single-shot mode, using BMOM bit in the HRTIM_BMCR register. The continuous mode is enabled when BMOM = 1. The burst operation is maintained until BMSTAT bit in HRTIM_BMCR is reset to terminate it.

In single-shot mode (BMOM = 0), the idle sequence is executed once, following the burst mode trigger, and the normal timer operation is resumed immediately after.

The duration of the idle and run periods is defined with a burst mode counter and 2 registers. The HRTIM_BMCMPR register defines the number of counts during which the selected timer(s) are in an idle state (idle period). HRTIM_BMPER defines the overall burst mode period (sum of the idle and run periods). Once the initial burst mode trigger has occurred, the idle period length is HRTIM_BMCMPR+1, the overall burst period is HRTIM_BMPER+1.

Note: The burst mode period must not be less than or equal to the deadtime duration defined with DTRx[8:0] and DTFx[8:0] bit fields.

The counters of the timing units and the master timer can be stopped and reset during the burst mode operation. HRTIM_BMCR holds 6 control bits for this purpose: MTBM (master) and TABM..TEBM for timer A..E.

When MTBM or TxBM bit is reset, the counter clock is maintained. This allows to keep a phase relationship with other timers in multiphase systems, for instance.

When MTBM or TxBM bit is set, the corresponding counter is stopped and maintained in reset state during the burst idle period. This allows to have the timer restarting a full period when exiting from idle. If SYNCSRC[1:0] = 00 or 10 (synchronization output on the master

start or timer A start), a pulse is sent on the HRTIM_SCOUT output when exiting the burst mode.

Note: TxBM bit must not be set when the balanced idle mode is active (DLYPRT[1:0] = 0x11).

Burst mode clock

The burst mode controller counter can be clocked by several sources, selected with BMCLK[3:0] bits in the HRTIM_BMCR register:

- BMCLK[3:0] = 0000 to 0101: master timer and TIMA..E reset/roll-over events. This allows to have burst mode idle and run periods aligned with the timing unit counting period (both in free-running and counter reset mode).
- BMCLK[3:0] = 0110 to 1001: The clocking is provided by the hrtim_bm_ck[4:1] inputs connected to general purpose timers, as in *Table 381*. In this case, the burst mode idle and run periods are not necessarily aligned with timing unit counting period. A pulse on the output may be interrupted, resulting in a waveform with modified duty cycle for instance.
- BMCLK[3:0] = 1010: The f_{HRTIM} clock prescaled by a factor defined with BMPRSC[3:0] bits in HRTIM_BMCR register. In this case, the burst mode idle and run periods are not necessarily aligned with the timing unit counting period. A pulse on the output may be interrupted, resulting in a waveform with a modified duty cycle, for instance.

The pulse width on TIMx OC output must be at least N f_{HRTIM} clock cycles long to be detected by the HRTIM burst mode controller.

Burst mode triggers

To trigger the burst operation, 32 sources are available and are selected using the HRTIM_BMTRGR register:

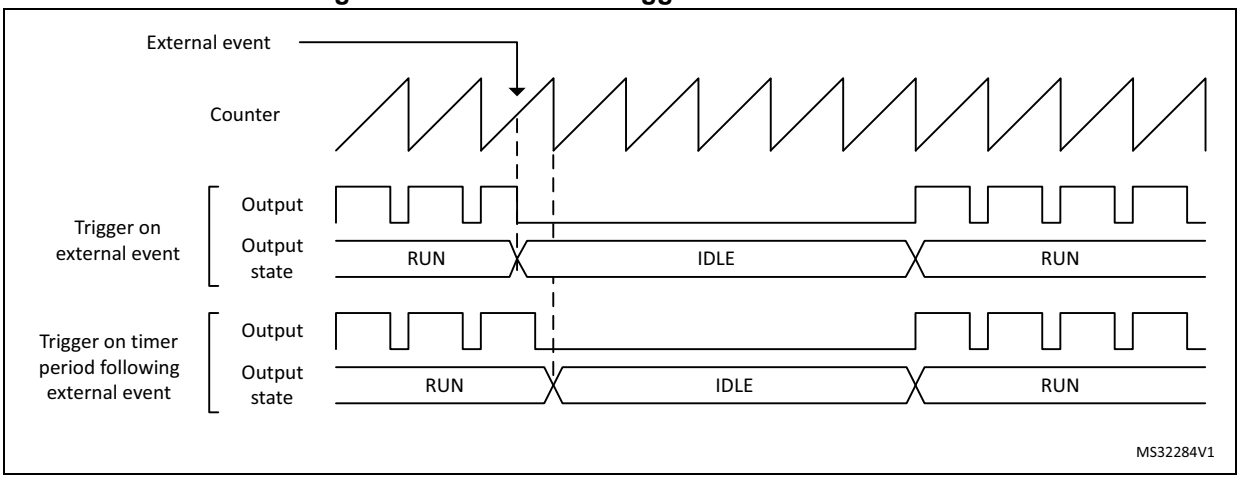
- Software trigger (set by software and reset by hardware)
- 6 master timer events: repetition, reset/roll-over, compare 1 to 4
- 5 x 4 events from timers A..F: repetition, reset/roll-over, compare 1 and 2
- External event 7 (including TIMA event filtering) and 8 (including TIMD event filtering)
- Timer A period following external event 7 (including TIMA event filtering)
- Timer D period following external event 8 (including TIMD event filtering)
- An on-chip event on the hrtim_bm_trg input (connected to the general-purpose timer TRGO output), refer to *Table 379* for details.

These sources can be combined to have multiple concurrent triggers.

Burst mode is not retriggerable. In continuous mode, new triggers are ignored until the burst mode is terminated, while in single-shot mode, the triggers are ignored until the current burst completion including run periods (HRTIM_BMPER+1 cycles). This is also valid for software trigger (the software bit is reset by hardware even if it is discarded).

Figure 392 shows how the burst mode is started in response to an external event, either immediately or on the timer period following the event.

Figure 392. Burst mode trigger on external event



For TAEEV7 and TDEEV8 combined triggers (trigger on a timer period following an external event), the external event detection is always active, regardless of the burst mode programming and the on-going burst operation:

- When the burst mode is enabled (BME=1) or the trigger is enabled (TAEEV7 or TDEEV8 bit set in the BMTRG register) in between the external event and the timer period event, the burst is triggered.
- The single-shot burst mode is re-triggered even if the external event occurs before the burst end (as long as the corresponding period happens after the burst).

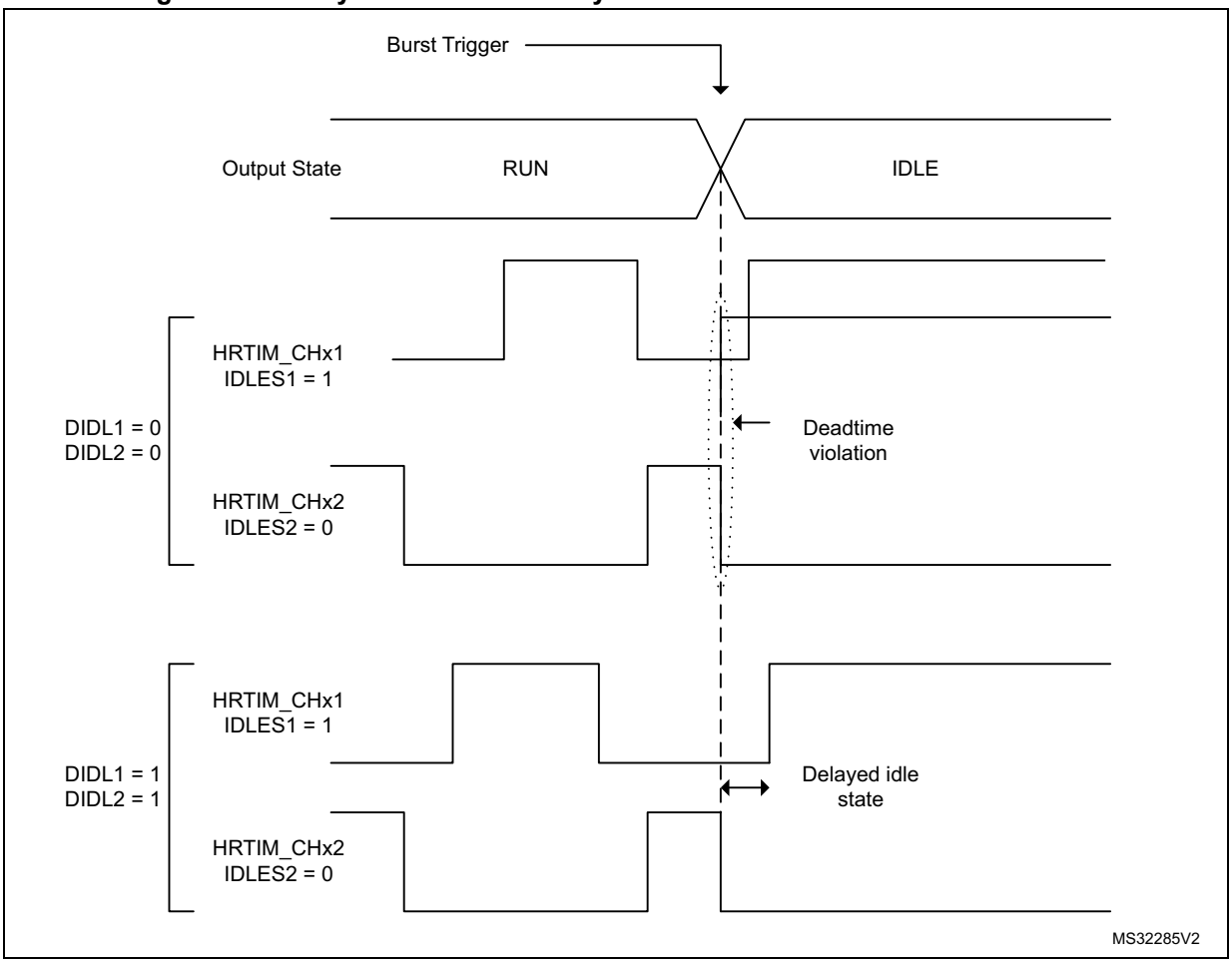
Note: TAEEV7 and TDEEV8 triggers are valid only after a period event. If the counter is reset before the period event, the pending hrtim_eev7 or hrtim_eev8 event is discarded.

Burst mode delayed entry

By default, the outputs are taking their idle level (as per IDLES1 and IDLES2 setting) immediately after the burst mode trigger.

It is also possible to delay the burst mode entry and force the output to an inactive state during a programmable period before the output takes its idle state. This is useful when driving two complementary outputs, one of them having an active idle state, to avoid a deadtime violation as shown in *Figure 393*. This prevents any risk of shoot through current in half-bridges, but causes a delayed response to the burst mode entry.

Figure 393. Delayed burst mode entry with deadtime enabled and IDLESx = 1



The delayed burst entry mode is enabled with DIDLx bit in the HRTIM_OUTxR register (one enable bit per output). It forces a deadtime insertion before the output takes its idle state. Each TIMx output has its own deadtime value:

- DTRx[8:0] on output 1 when DIDL1 = 1
- DTFx[8:0] on output 2 when DIDL2 = 1

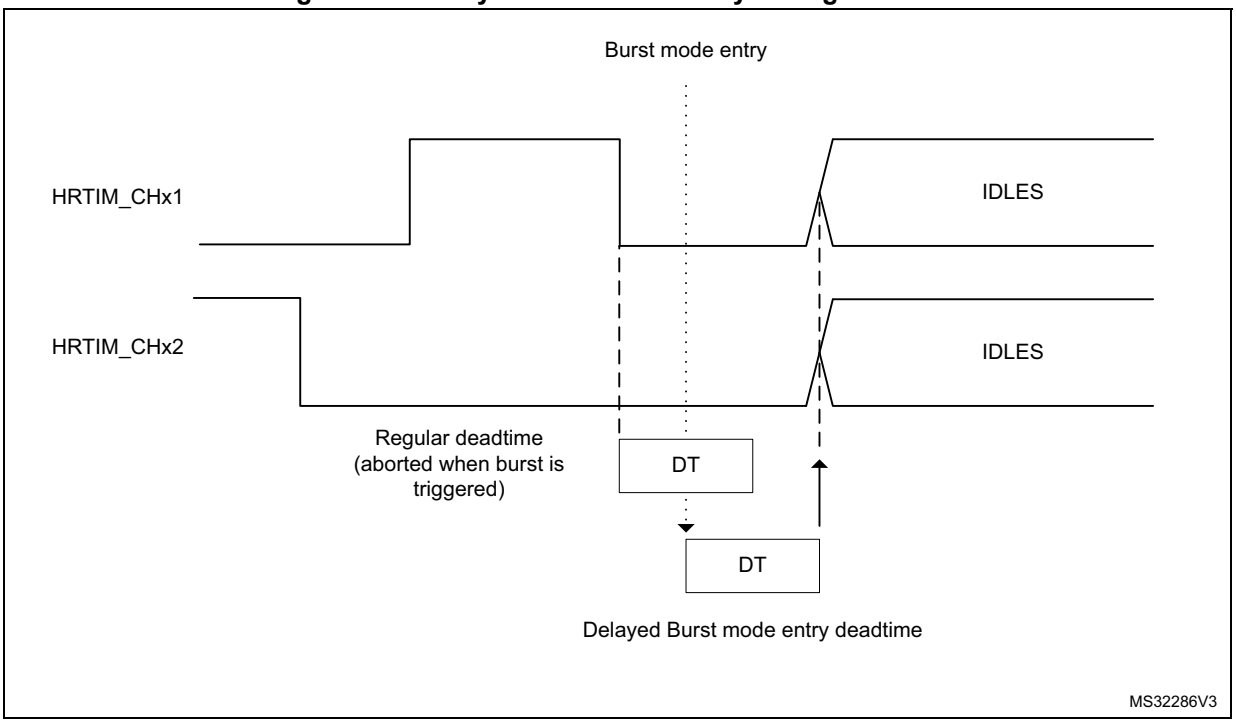
DIDLx bits can be set only if one of the outputs has an active idle level during the burst mode (IDLES = 1) and only when positive deadtimes are used (SDTR/SDTF set to 0).

Note: The delayed burst entry mode uses deadtime generator resources. Consequently, when any of the 2 DIDLx bits is set and the corresponding timing unit uses the deadtime insertion (DTEN bit set in HRTIM_OUTxR), it is not possible to use the timerx output 2 as a filter for external events (Tx2 filtering signal is not available).

When durations defined by DTRx[8:0] and DTFx[8:0] are lower than 3 f_{HRTIM} clock cycle periods, the limitations related to the narrow pulse management listed in *Section 34.3.7* must be applied.

When the burst mode entry arrives during the regular deadtime, it is aborted and a new deadtime is re-started corresponding to the inactive period, as in *Figure 394*.

Figure 394. Delayed burst mode entry during deadtime



Burst mode exit

The burst mode exit is either forced by software (in continuous mode) or once the idle period is elapsed (in single-shot mode). In both cases, the counter is re-started immediately (if it was hold in a reset state with MTBM or TxBM bit = 1), but the effective output state transition from the idle to active mode only happens after the programmed set/reset event.

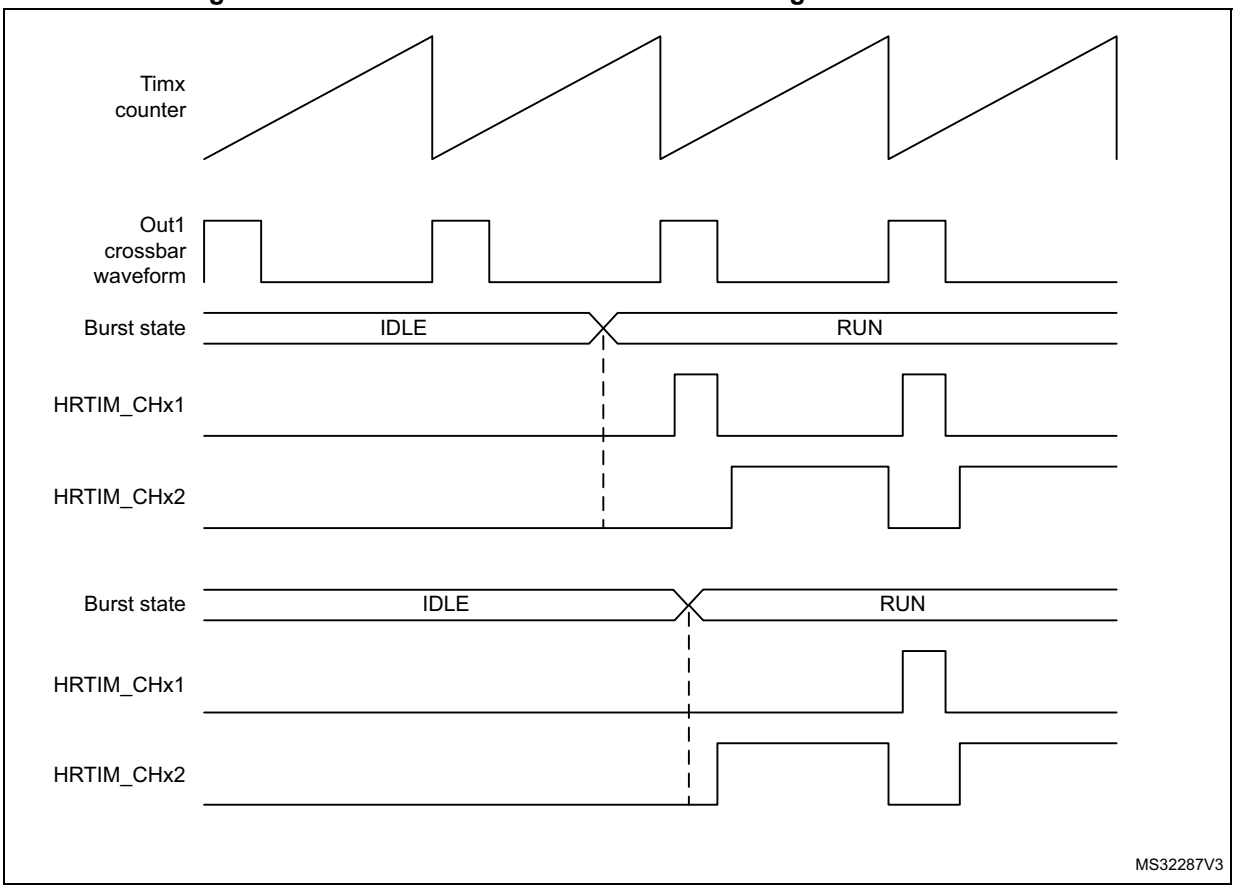
A burst period interrupt is generated in single-shot and continuous modes when BMPERIE enable bit is set in the HRTIM_IER register. This interrupt can be used to synchronize the burst mode exit with a burst period in continuous burst mode.

Figure 395 shows how a normal operation is resumed when the deadtime is enabled. Although the burst mode exit is immediate, this is only effective on the first set event on any of the complementary outputs.

Two different cases are presented:

1. The burst mode ends while the signal is inactive on the crossbar output waveform. The active state is resumed on Tx1 and Tx2 on the set event for the Tx1 output, and the Tx2 output does not take the complementary level on burst exit.
2. The burst mode ends while the crossbar output waveform is active: the activity is resumed on the set event of Tx2 output, and Tx1 does not take the active level immediately on burst exit.

Figure 395. Burst mode exit when the deadtime generator is enabled



The behavior described above is slightly different when the push-pull mode is enabled. The push-pull mode forces an output reset at the beginning of the period if the output is inactive, or symmetrically forces an active level if the output was high during the preceding period.

Consequently, an output with an active idle state can be reset at the time the burst mode is exited even if no transition is explicitly programmed. For symmetrical reasons, an output can be set at the time the burst mode is exited even if no transition is explicitly programmed, in case it was active when it entered in idle state.

Burst mode registers preloading and update

BMPREN bit (burst mode preload enable) allows to have the burst mode compare and period registers preloaded (HRTIM_BMCMP and HRTIM_BMPER).

When BMPREN is set, the transfer from preload to active register happens:

- When the burst mode is enabled (BME = 1),
- At the end of the burst mode period.

A write into the HRTIM_BMPER period register disables the update temporarily, until the HRTIM_BMCMP compare register is written, to ensure the consistency of the two registers when they are modified.

If the compare register only needs to be changed, a single write is necessary. If the period only needs to be changed, it is also necessary to re-write the compare to have the new values taken into account.

When BMPREN bit is reset, the write access into BMCMPR and BMPER directly updates the active register. In this case, it is necessary to consider when the update is done during the overall burst period, for the 2 cases below:

a) Compare register update

If the new compare value is above the current burst mode counter value, the new compare is taken into account in the current period.

If the new compare value is below the current burst mode counter value, the new compare is taken into account in the next burst period in continuous mode, and ignored in single-shot mode (no compare match occurs and the idle state lasts until the end of the idle period).

b) Period register update

If the new period value is above the current burst mode counter value, the change is taken into account in the current period.

Note: If the new period value is below the current burst mode counter value, the new period is not taken into account, the burst mode counter overflows (at 0xFFFF) and the change is effective in the next period. In single-shot mode, the counter rolls over at 0xFFFF and the burst mode re-starts for another period up to the new programmed value.

Burst mode emulation using a compound register

The burst mode controller only controls one or a set of timers for a single converter. When the burst mode is necessary for multiple independent timers, it is possible to emulate a simple burst mode controller using the DMA and the HRTIM_CMP1CxR compound register, which holds aliases of both the repetition and the compare 1 registers.

This is applicable to a converter which only requires a simple PWM (typically a buck converter), where the duty cycle only needs to be updated. In this case, the CMP1 register is used to reset the output (and define the duty cycle), while it is set on the period event.

In this case, a single 32-bit write access in CMP1CxR is sufficient to define the duty cycle (with the CMP1 value) and the number of periods during which this duty cycle is maintained (with the repetition value). To implement a burst mode, it is then only necessary to transfer by DMA (upon repetition event) two 32-bit data in continuous mode, organized as follows:

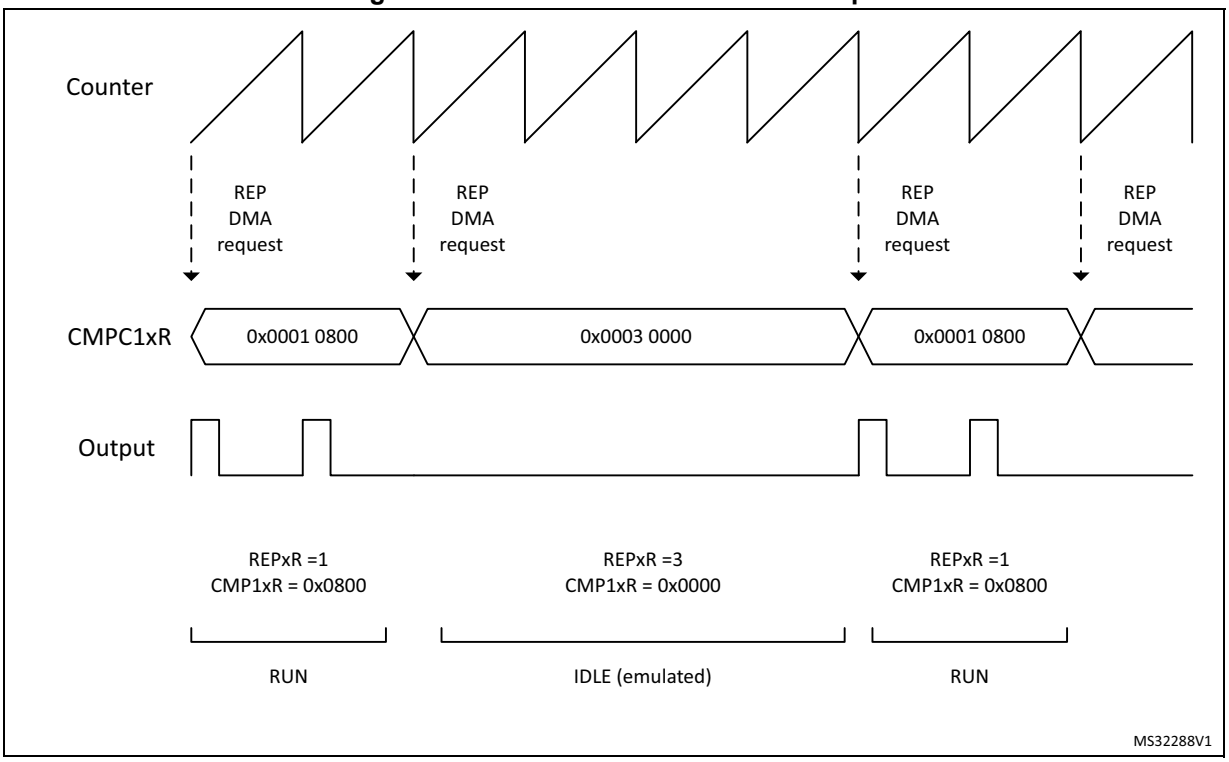
CMPC1xR = {REP_Run; CMP1 = Duty_Cycle}, {REP_Idle; CMP1 = 0}

For instance, the values:

- {0x0003 0000}: CMP1 = 0 for 3 periods
- {0x0001 0800}: CMP1 = 0x0800 for 1 period

provide a burst mode with 2 periods active every 6 PWM periods, as shown in *Figure 396*.

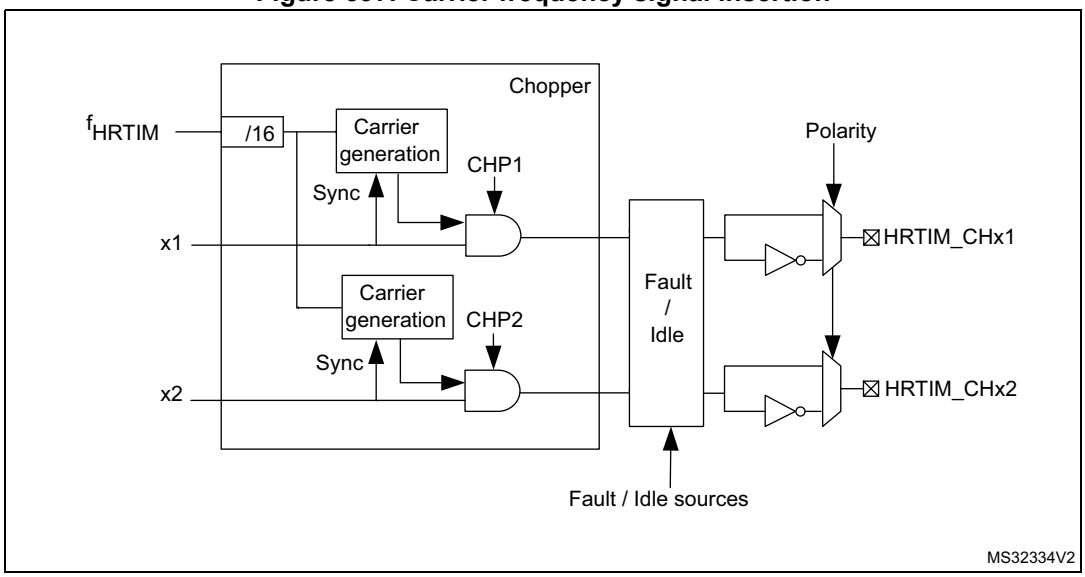
Figure 396. Burst mode emulation example



34.3.16 Chopper

A high-frequency carrier can be added on top of the timing unit output signals to drive isolation transformers. This is done in the output stage before the polarity insertion, as shown in *Figure 397*, using CHP1 and CHP2 bits in the HRTIM_OUTxR register, to enable chopper on outputs 1 and 2, respectively.

Figure 397. Carrier frequency signal insertion



The chopper parameters can be adjusted using the HRIM_CHPxR register, with the possibility to define a specific pulsewidth at the beginning of the pulse, to be followed by a carrier frequency with programmable frequency and duty cycle, as in *Figure 398*.

CARFRQ[3:0] bits define the frequency, ranging from 1.172 to 18.75 MHz (for fhrtim = 300 MHz) following the formula $F_{CHPFRQ} = f_{HRTIM} / (16 \times (CARFRQ[3:0]+1))$.

The duty cycle can be adjusted by 1/8 step with CARDTY[2:0], from 0/8 up to 7/8 duty cycle. When CARDTY[2:0] = 000 (duty cycle = 0/8), the output waveform only contains the starting pulse following the rising edge of the reference waveform, without any added carrier.

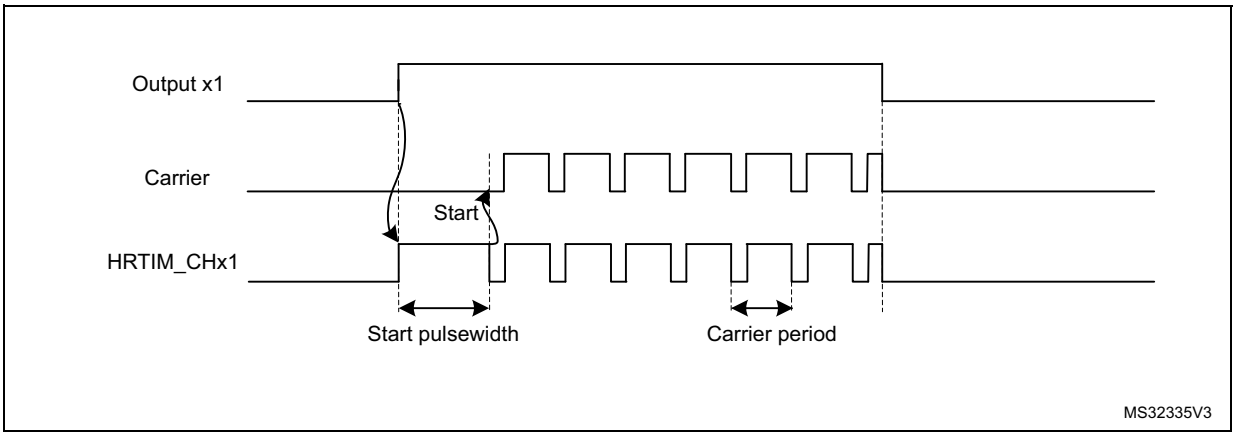
The pulsewidth of the initial pulse is defined using the STRPW[3:0] bit field as follows: $t1STPW = (STRPW[3:0]+1) \times 16 \times t_{HRTIM}$ and ranges from 53 ns to 853 ns (for f_{HRTIM}=300 MHz).

The carrier frequency parameters are defined based on the f_{HRTIM} frequency, and are not dependent on the CKPSC[2:0] setting.

In chopper mode, the carrier frequency and the initial pulsewidth are combined with the reference waveform using an AND function. A synchronization is performed at the end of the initial pulse to have a repetitive signal shape.

The chopping signal is stopped at the end of the output waveform active state, without waiting for the current carrier period to be completed. It can thus contain shorter pulses than programmed.

Figure 398. HRTIM outputs with Chopper mode enabled



Note: CHP1 and CHP2 bits must be set prior to the output enable done with TxyOEN bits in the HRTIM_OENR register.

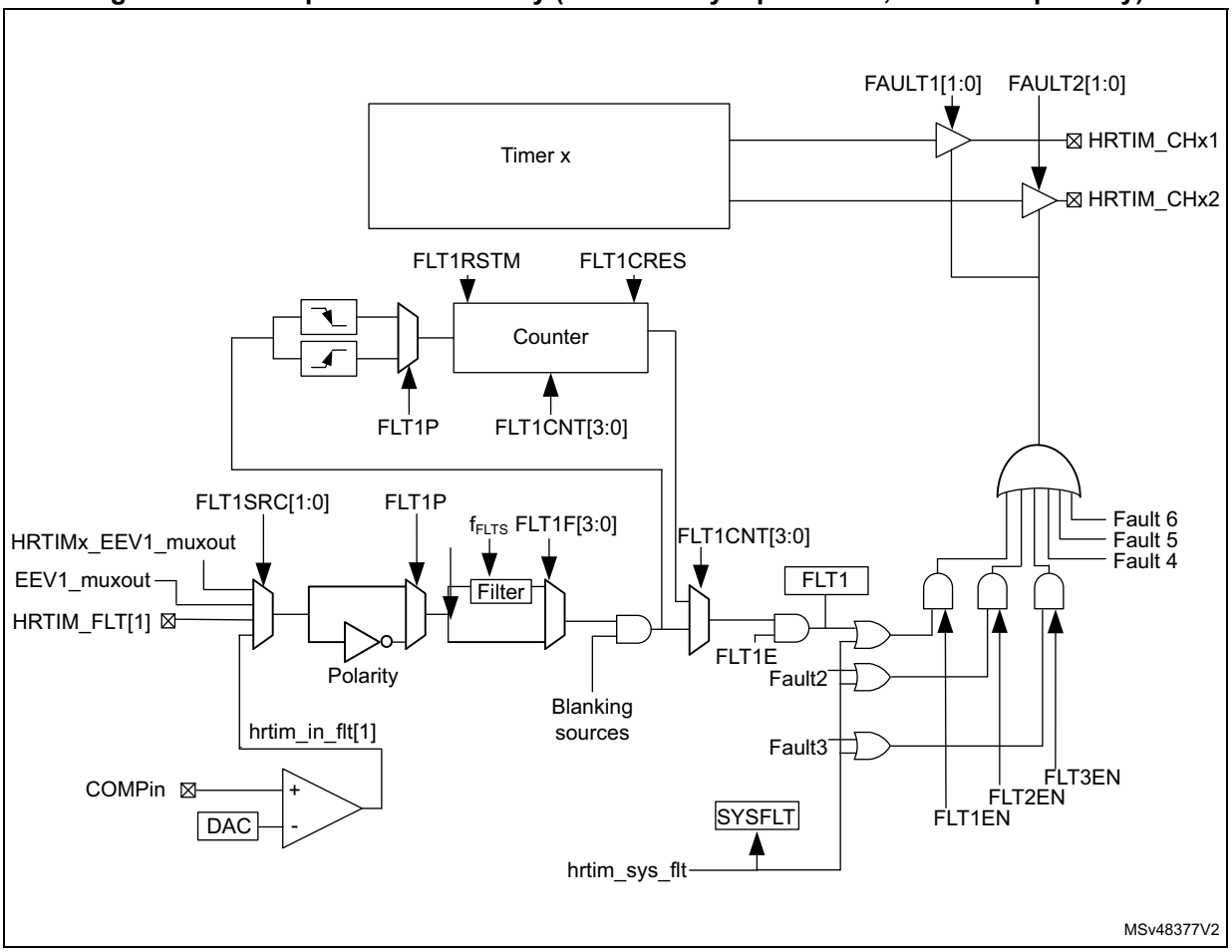
CARFRQ[2:0], CARDTY[2:0] and STRPW[3:0] bit fields cannot be modified while the chopper mode is active (at least one of the two CHPx bits is set).

34.3.17 Fault protection

The HRTIM has a versatile fault protection circuitry to disable the outputs in case of an abnormal operation. Once a fault has been triggered, the outputs take a predefined safe state. This state is maintained until the output is re-enabled by software. In case of a permanent fault request, the output remains in its fault state, even if the software attempts to re-enable them, until the fault source disappears.

The HRTIM has 6 FAULT input channels; all of them are available and can be combined for each of the 6 timing units, as shown in *Figure 399*.

Figure 399. Fault protection circuitry (FAULT1 fully represented, FAULT2..6 partially)



Each fault channel is fully configurable using HRTIM_FLTINR1 and HRTIM_FLTINR2 registers before being routed to the timing units. FLTxSRC FLTxSRC[1:0] bit selects the source of the fault signal, that can be either a digital input or an internal event (built-in comparator output).

The available sources for each of the six faults channels are summarized in the HRTIM fault input tables of *Chapter 5: Device configuration*.

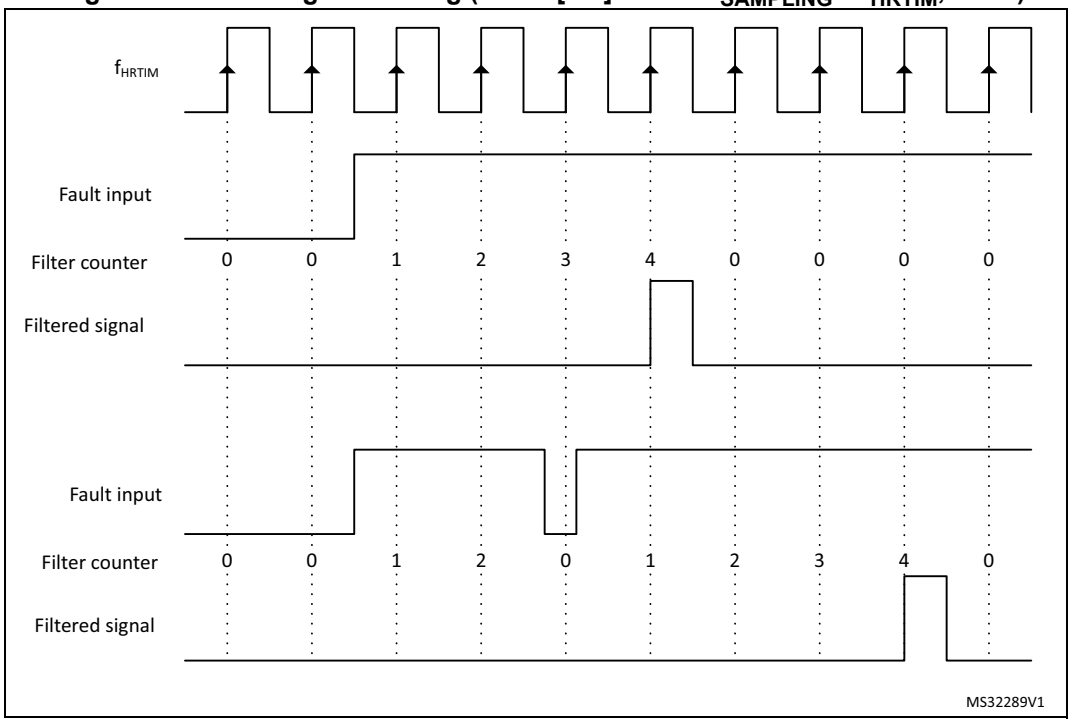
The EEVx_muxout event and the HRTIMx_EEVx_muxout event mentioned in HRTIM fault inputs tables of *Chapter 5: Device configuration* are taken after the hrtim_eevx[4:1] input multiplexer controlled by the EExSRC[1:0]bits. Refer to *Figure 371* for details.

The polarity of the signal can be selected to define the active level, using the FLTxP polarity bit in HRTIM_FLTINRx registers. If FLTxP = 0, the signal is active at low level; if FLTxP = 1, it is active when high.

The fault information can be filtered after the polarity setting. If FLTxF[3:0] bit field is set to 0000, the signal is not filtered and acts asynchronously, independently from the f_{HRTIM} clock. For all other FLTxF[3:0] bit field values, the signal is digitally filtered. The digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event.

Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected). *Figure 400* shows how a spurious fault signal is filtered.

Figure 400. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4)



The filtering period ranges from 2 cycles of the f_{HRTIM} clock up to 8 cycles of the f_{FLTS} clock divided by 32. f_{FLTS} is defined using FLTSD[1:0] bits in the HRTIM_FLTINR2 register. *Table 402* summarizes the sampling rate and the filter length. A jitter of 1 sampling clock period must be subtracted from the filter length to take into account the uncertainty due to the sampling and have the effective filtering.

Table 402. Sampling rate and filter length versus FLTFxF[3:0] and clock setting

-	f_{FLTS} vs FLTSD[1:0]				Filter length for f_{HRTIM} = 300 MHz	
FLTFxF[3:0]	**00**	**01**	**10**	**11**	**Min**	**Max**
0001, 0010, 0011	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}, N =2 6.7 ns	f_{HRTIM}, N =8 26.7 ns
0100, 0101	f_{HRTIM} /2	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /2, N = 6 40 ns	f_{HRTIM} /16, N = 8 427 ns
0110, 0111	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /4, N = 6 80 ns	f_{HRTIM} /32, N = 8 853 ns
1000, 1001	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /8, N = 6 160 ns	f_{HRTIM} /64, N = 8 1.71 µs
1010, 1011, 1100	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /16, N = 5 267 ns	f_{HRTIM} /128, N = 8 3.41 µs
1101, 1110, 1111	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /256	f_{HRTIM} /32, N = 5 533 ns	f_{HRTIM} /256, N = 8 6.82 µs

Fault blanking and event counting

The fault inputs can be temporarily disabled to blank spurious fault events. The blanking sources are listed in *Table 403*.

Table 403. Fault input blanking events

-	FLTxBLKS = 0, reset-aligned window		FLTxBLKS = 1 moving window	
Fault input	**Blanking window start**	**Blanking window end**	**Blanking window start**	**Blanking window end**
hrtim_flt1[4:1]	Timer A reset/roll-over	Timer A CMP3 event	Timer A CMP4 event	Timer A CMP3 event
hrtim_flt2[4:1]	Timer B reset/roll-over	Timer B CMP3 event	Timer B CMP4 event	Timer B CMP3 event
hrtim_flt3[4:1]	Timer C reset/roll-over	Timer C CMP3 event	Timer C CMP4 event	Timer C CMP3 event
hrtim_flt4[4:1]	Timer D reset/roll-over	Timer D CMP3 event	Timer D CMP4 event	Timer D CMP3 event
hrtim_flt5[4:1]	Timer E reset/roll-over	Timer E CMP3 event	Timer E CMP4 event	Timer E CMP3 event
hrtim_flt6[4:1]	Timer F reset/roll-over	Timer F CMP3 event	Timer F CMP4 event	Timer F CMP3 event

A fault counter also allows to discard multiple spurious fault events and define an acceptance criteria.

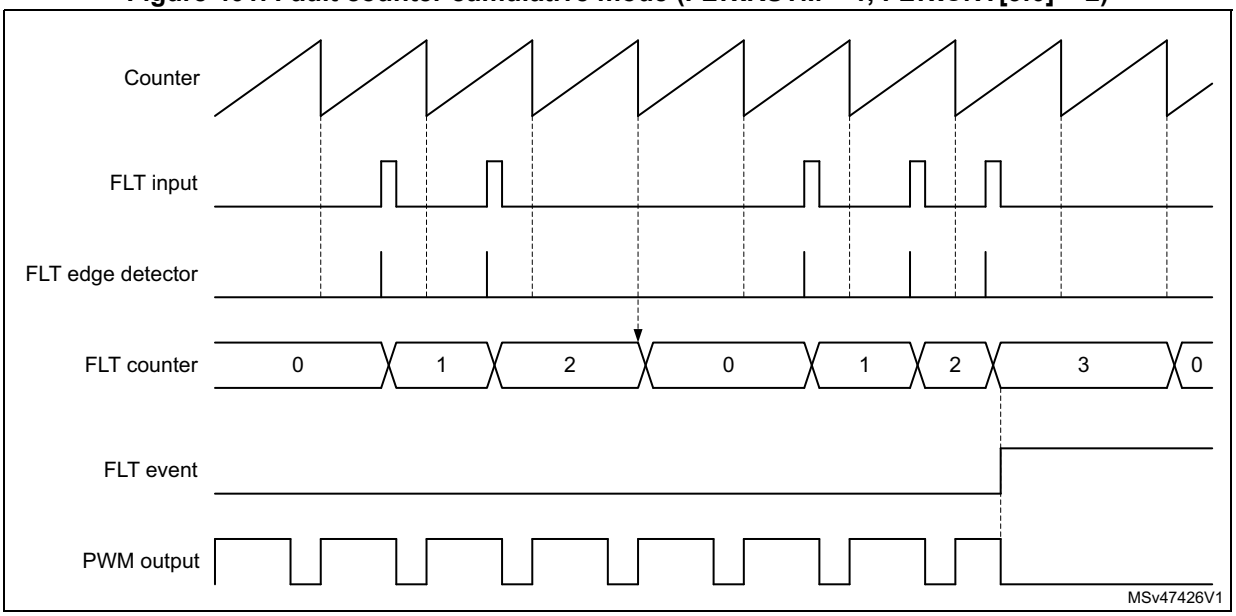
The FLTxCNT[3:0] bit field selects the FAULTx counter threshold. A fault is considered valid when the number of events is equal to the FLTxCNT[3:0] value.

This FLTxRSTM selects the FAULTx counter reset mode:

- 0: fault counter hardware reset on reset / roll-over event, as per *Table 404*.
- 1: fault counter reset on each reset / roll-over event only if no event occurs during last counting period, as shown in *Figure 401*.

The fault counter can be reset by software with the FLTxCRES bit at anytime.

Figure 401. Fault counter cumulative mode (FLTxRSTM = 1, FLTxCNT[3:0] = 2)



A given FLTx input counter can be reset by a single source. *Table 404* indicates which timer unit is associated with a given fault. This does not prevent to have a fault line shared by multiple timer (for example, FLT1 with event counter enabled, acting on timer A, timer B and timer C simultaneously).

Table 404. Fault 1..6 counter reset source

Fault Input	Fault counter reset source
hrtim_flt1[4:1]	Timer A reset/roll-over
hrtim_flt2[4:1]	Timer B reset/roll-over
hrtim_flt3[4:1]	Timer C reset/roll-over
hrtim_flt4[4:1]	Timer D reset/roll-over
hrtim_flt5[4:1]	Timer E reset/roll-over
hrtim_flt6[4:1]	Timer F reset/roll-over

System fault input (hrtim_sys_flt)

This fault is provided by the MCU Class B circuitry (see the system configuration controller (SYSCFG) section for details) and corresponds to a system fault coming from the Fault Collection and Control Unit (FCCU).

This input overrides the FAULT inputs and disables all outputs having FAULTy[1:0] = 01, 10, 11.

For each FAULT channel, a write-once FLTxLCK bit in the HRTIM_FLTxR register allows to lock FLTxE, FLTxP, FLTxSRC, FLTxF[3:0] bits (it renders them read-only), for functional safety purpose. If enabled, the fault conditioning set-up is frozen until the next HRTIM or system reset.

Once the fault signal is conditioned as explained above, it is routed to the timing units. For any of them, the 6 fault channels are enabled using bits FLT1EN to FLT6EN in the HRTIM_FLTxR register, and they can be selected simultaneously (the sysfault is automatically enabled as long as the output is protected by the fault mechanism). This allows to have, for instance:

- One fault channel simultaneously disabling several timing units.
- Multiple fault channels being ORed to disable a single timing unit.

A write-once FLTLCK bit in the HRTIM_FLTxR register allows to lock FLTxEN bits (it renders them read-only) until the next reset, for functional safety purpose. If enabled, the timing unit fault-related set-up is frozen until the next HRTIM or system reset.

For each of the timers, the output state during a fault is defined with FAULT1[1:0] and FAULT2[1:0] bits in the HRTIM_OUTxR register (refer to *Section 34.3.14*).

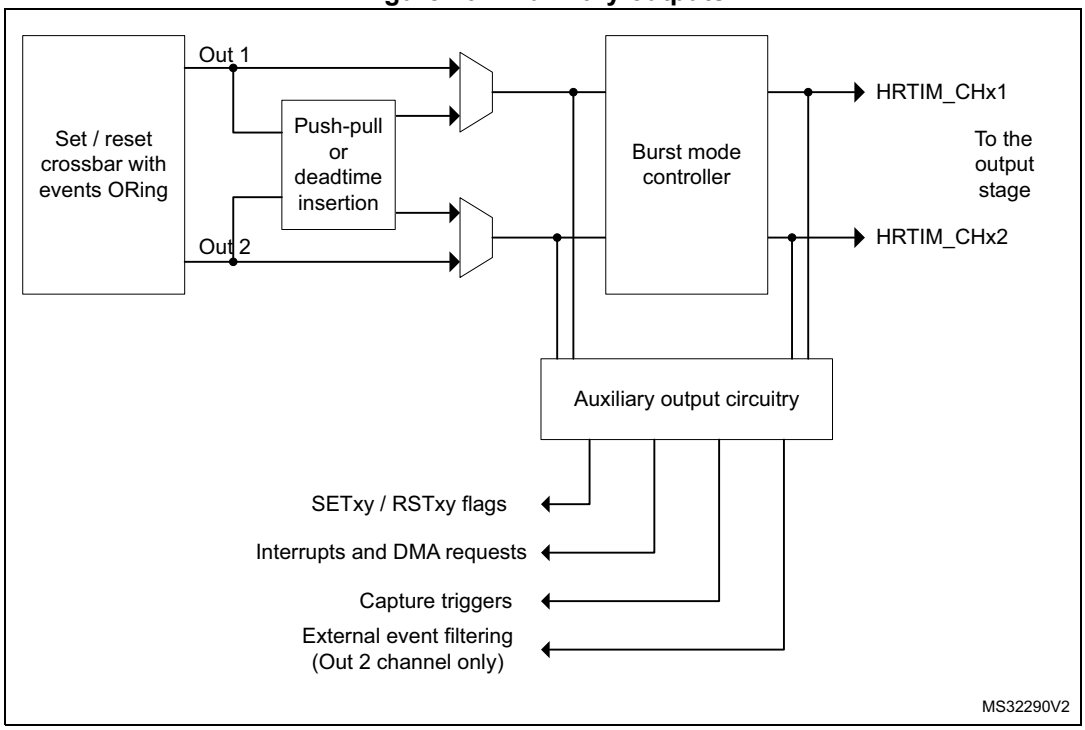
34.3.18 Auxiliary outputs

Timer A to E have auxiliary outputs in parallel with the regular outputs going to the output stage. They provide the following internal status, events and signals:

- SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests.
- Capture triggers upon output set/reset.
- External event filters following a Tx2 output copy (refer to details in *Section 34.3.9*).

The auxiliary outputs are taken either before or after the burst mode controller, depending on the HRTIM operating mode. An overview is given in *Figure 402*.

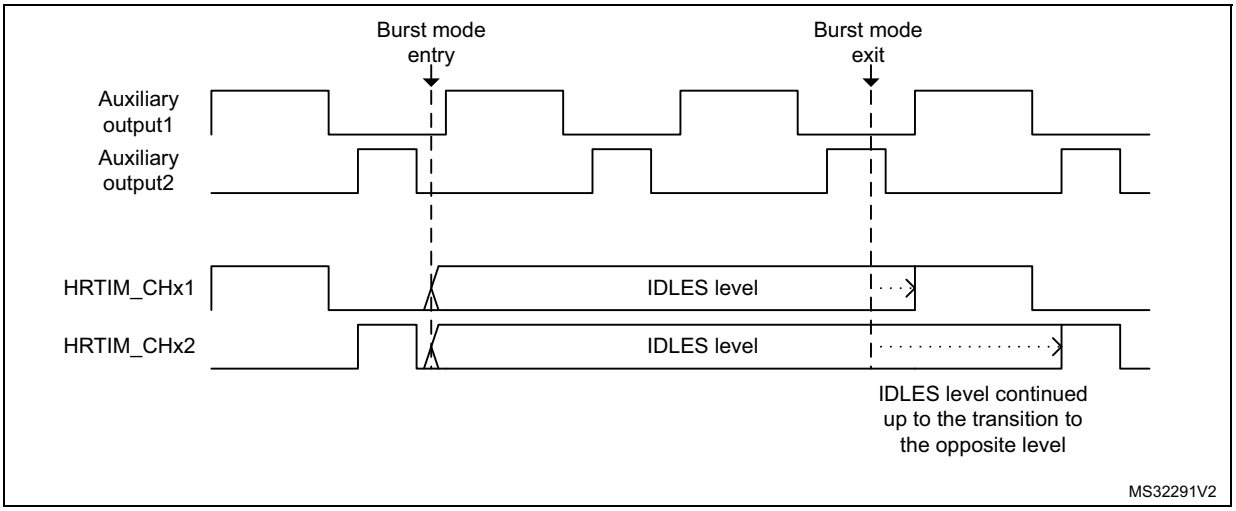
Figure 402. Auxiliary outputs



By default, the auxiliary outputs are copies of outputs Tx1 and Tx2. The exceptions are:

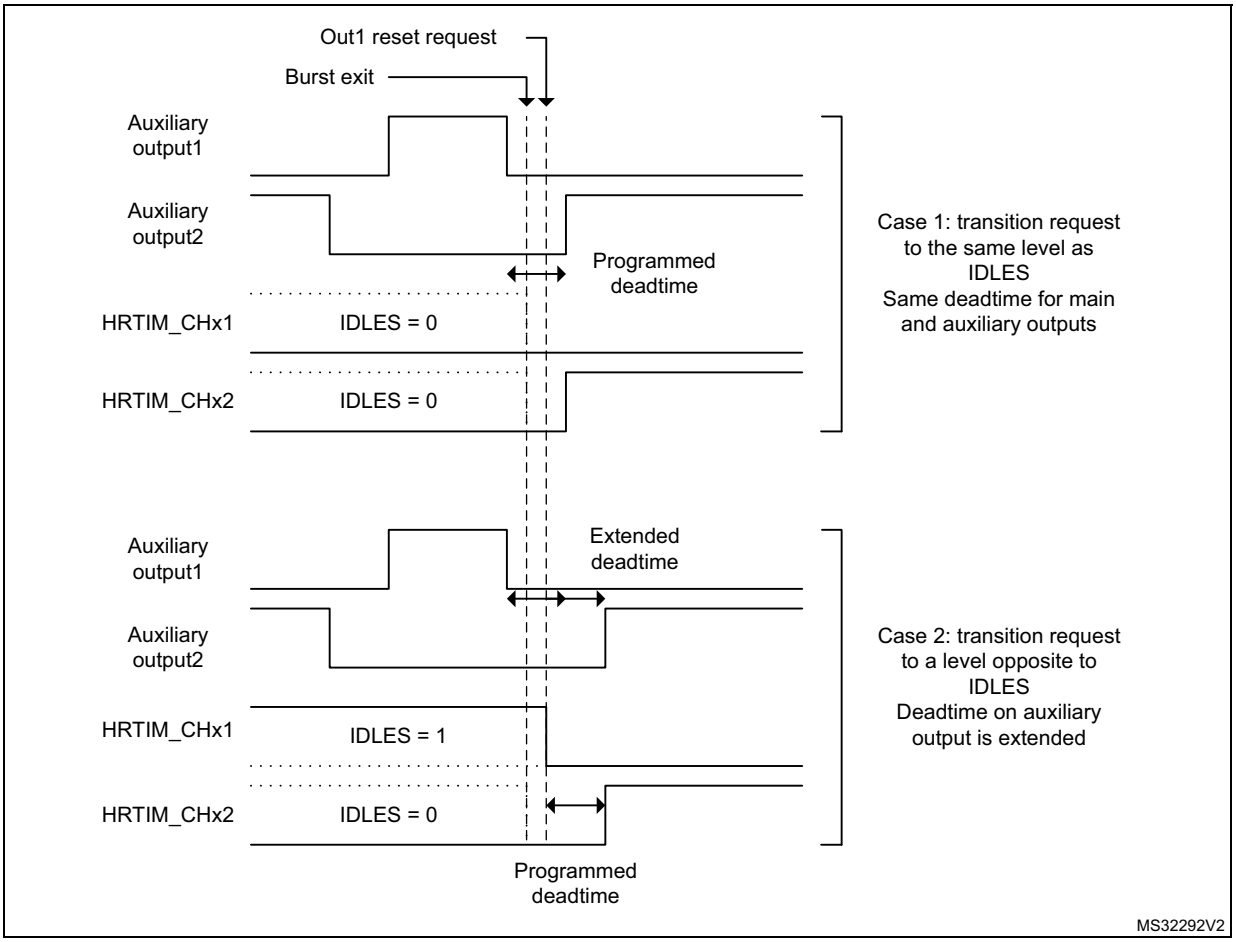
- The delayed idle and the balanced idle protections, when the deadtime is disabled (DTEN = 0). When the protection is triggered, the auxiliary outputs are maintained and follow the signal coming out of the crossbar. On the contrary, if the deadtime is enabled (DTEN = 1), both main and auxiliary outputs are forced to an inactive level.
- The burst mode (TCEN=1, IDLEMx=1); there are 2 cases:
  a) If DTEN=0 or DIDLx=0, the auxiliary outputs are not affected by the burst mode entry and continue to follow the reference signal coming out of the crossbar (refer to *Figure 403*).
  b) If the deadtime is enabled (DTEN=1) together with the delayed burst mode entry (DIDLx=1), the auxiliary outputs have the same behavior as the main outputs. They are forced to the IDLES level after a deadtime duration, then they keep this level during all the burst period. When the burst mode is terminated, the IDLES level is maintained until a transition occurs to the opposite level, similarly to the main output.

Figure 403. Auxiliary and main outputs during burst mode (DIDLx = 0)



The signal on the auxiliary output can be slightly distorted when exiting from the burst mode or when re-enabling the outputs after a delayed protection, if this happens during a deadtime. In this case, the deadtime applied to the auxiliary outputs is extended so that the deadtime on the main outputs is respected. *Figure 404* gives some examples.

Figure 404. Deadtime distortion on auxiliary output when exiting burst mode



34.3.19 Synchronizing the HRTIM with other timers or HRTIM instances

The HRTIM provides options for synchronizing multiple HRTIM instances, as a master unit (generating a synchronization signal) or as a slave (waiting for a trigger to be synchronized). This feature can also be used to synchronize the HRTIM with other timers, either external or on-chip. The synchronization circuitry is controlled inside the master timer.

Synchronization output

This section explains how the HRTIM must be configured to synchronize external resources and act as a master unit.

Four events can be selected as the source to be sent to the synchronization output. This is done using SYNCSRC[1:0] bits in the HRTIM_MCR register, as follows:

- 00: master timer start:
  This event is generated when MCEN bit is set or when the timer is re-started after having reached the period value in single-shot mode. It is also generated on a reset which occurs during the counting (when CONT or RETRIG bits are set).
- 01: master timer compare 1 event.
- 10: timer A start:
  This event is generated when TACEN bit is set or when the counter is reset and re-starts counting in response to this reset. The following counter reset events are not propagated to the synchronization output: counter roll-over in continuous mode, and discarded reset request in single-shot non-retriggerable mode. The reset is only taken into account when it occurs during the counting (CONT or RETRIG bits are set).
- 11: timer A compare 1 event.

SYNCOUT[1:0] bits in the HRTIM_MCR register specify how the synchronization event is generated.

The synchronization pulses are generated on the HRTIM_SCOUT output pin, with SYNCOUT[1:0] = 1x. SYNCOUT[0] bit specifies the polarity of the synchronization signal. If SYNCOUT[0] = 0, the HRTIM_SCOUT pin has a low idle level and issues a positive pulse of 16 f_{HRTIM} clock cycles length for the synchronization). If SYNCOUT[0] = 1, the idle level is high and a negative pulse is generated.

Note: The synchronization pulse is followed by an idle level of 16 f_{HRTIM} *clock cycles during which any new synchronization request is discarded. Consequently, the maximum synchronization frequency is* $f_{HRTIM}/32$.

The idle level on the HRTIM_SCOUT pin is applied as soon as the SYNCOUT[1:0] bits are enabled (that is the bit field value is different from 00).

The synchronization output initialization procedure must be done prior to the configuration of the MCU outputs and counter enable, in the following order:

1. SYNCOUT[1:0] and SYNCSRC[1:0] bit field configuration in HRTIM_MCR
2. HRTIM_SCOUT pin configuration (see the General-purpose I/Os section)
3. Master or timer A counter enable (MCEN or TACEN bit set)

When the synchronization input mode is enabled and starts the counter (using SYNCSTRTM/SYNCSTRTx bits) simultaneously with the synchronization output mode (SYNCSRC[1:0] = 00 or 10), the output pulse is generated only when the counter is starting or is reset while running. Any reset request clearing the counter without causing it to start does not affect the synchronization output.

Synchronization input

The HRTIM can be synchronized by external sources, as per the programming of the SYNCIN[1:0] bits in the HRTIM_MCR register:

- 00: synchronization input is disabled.
- 01: another on-chip HRTIM instance.
- 10: the On-chip timer TRGO output connected to hrtim_in_sync[2] input (refer to *Table 379* for details).
- 11: a positive pulse on the HRTIM_SCIN input pin (hrtim_in_sync[3])

This bit field cannot be changed once the destination timer (master timer or timing unit) is enabled (MCEN and/or TxCEN bit set).

The HRTIM_SCIN input is rising-edge sensitive. The timer behavior is defined with the following bits present in HRTIM_MCR and HRTIM_TIMxCR registers (refer to *Table 405* for details):

- Synchronous start: the incoming signal starts the timer's counter (SYNCSTRTM and/or SYNCSTRTx bits set). TxCEN (MCEN) bits must be set to have the timer enabled and the counter ready to start. In continuous mode, the counter does not start until the synchronization signal is received.
- Synchronous reset: the incoming signal resets the counter (SYNCRSTM and/or SYNCRSTx bits set). This event decrements the repetition counter as any other reset event.

The synchronization events are taken into account only once the related counters are enabled (MCEN or TxCEN bit set). A synchronization request triggers a SYNC interrupt.

Note: A synchronized start event resets the counter if the current counter value is above the active period value.

The effect of the synchronization event depends on the timer operating mode, as summarized in *Table 405*.

Table 405. Effect of sync event versus timer operating modes

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot non-retriggerable	0	1	Start events are taken into account when the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be re-started with the synchronization). Any reset occurring during the counting is ignored (as during regular non-retriggerable mode).
	1	X	Reset events are starting the timer counting. They are taken into account only if the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. When multiple reset requests are selected (from HRTIM_SCIN and from internal events), only the first arriving request is taken into account.

Table 405. Effect of sync event versus timer operating modes (continued)

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot retriggerable	0	1	The counter start is effective only if the counter is not started or period is elapsed. Any synchronization event occurring after counter start has no effect. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be started by the synchronization). A reset occurring during counting is taken into account (as during regular retriggerable mode).
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM's timer counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.
Continuous mode	0	1	The timer is enabled (MCEN or TxCEN bit set) and is waiting for the synchronization event to start the counter. Any synchronization event occurring after the counter start has no effect (the counter can solely be started by the synchronization). A reset request clears the counter but does not start it.
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM's timer counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.

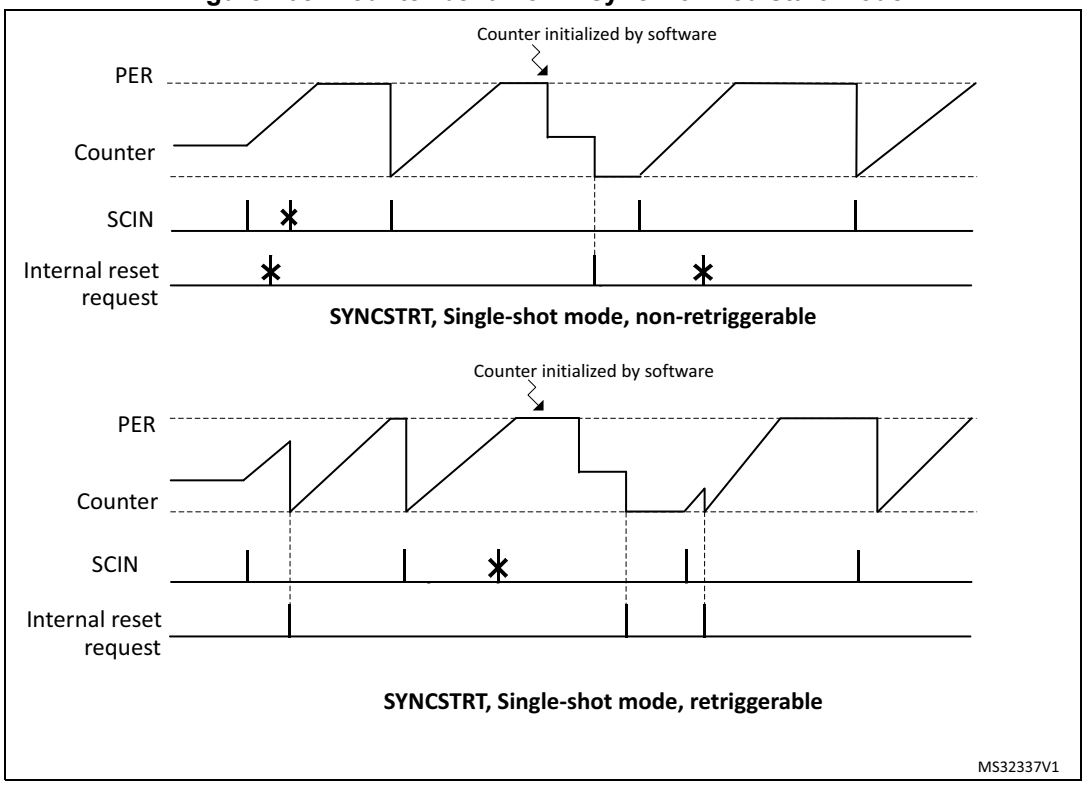
Note: When a synchronization reset event occurs within the same f_{HRTIM} *clock cycle as the period event, this reset is postponed to a programmed period event (since both events are causing a counter roll-over). This applies only when the high-resolution is active (*CKPSC[2:0] < 5).

When multiple HRTIMs are synchronously started, the HRTIM which initiates the synchronization is started 1 f_{HRTIM} *clock cycle after its SYNC output signal is sent. This allows to compensate for the internal propagation delay and to have the subsequent HRTIM timer cycle accurately synchronized.*

When multiple HRTIMs are synchronously reset, the slave timer is reset 1 f_{HRTIM} *clock cycle after the master request (internal propagation delay).*

Figure 405 presents how the synchronized start is done in single-shot mode.

Figure 405. Counter behavior in synchronized start mode



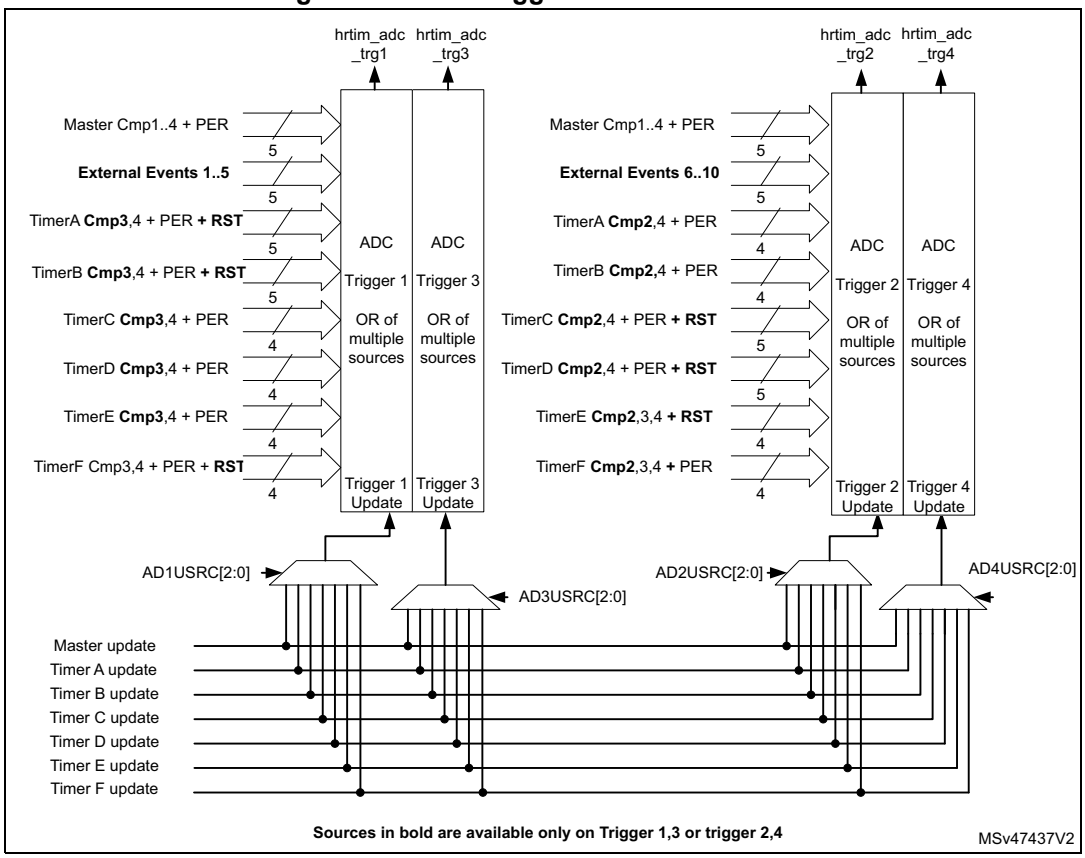
34.3.20 ADC triggers

The ADCs can be triggered by the master and the 6 timing units.

10 independent triggers are available for both the regular and the injected sequencers of the ADCs. The external events can be used as triggers. They are taken right after the conditioning defined in the HRTIM_EECRx registers, and are not depending on the EEFxR1 and EEFxR2 register settings.

Up to 32 events can be combined (ORed) for ADC triggers 1 to 4, in HRTIM_ADC1R to HRTIM_ADC4R registers, as shown in *Figure 406*. The ADC triggers 1/3 and 2/4 are using the same source set. A multiple triggering is possible within a single switching period by selecting several sources simultaneously. A typical use case is for a non-overlapping multiphase converter, where all phases can be sampled in a row using a single ADC trigger output.

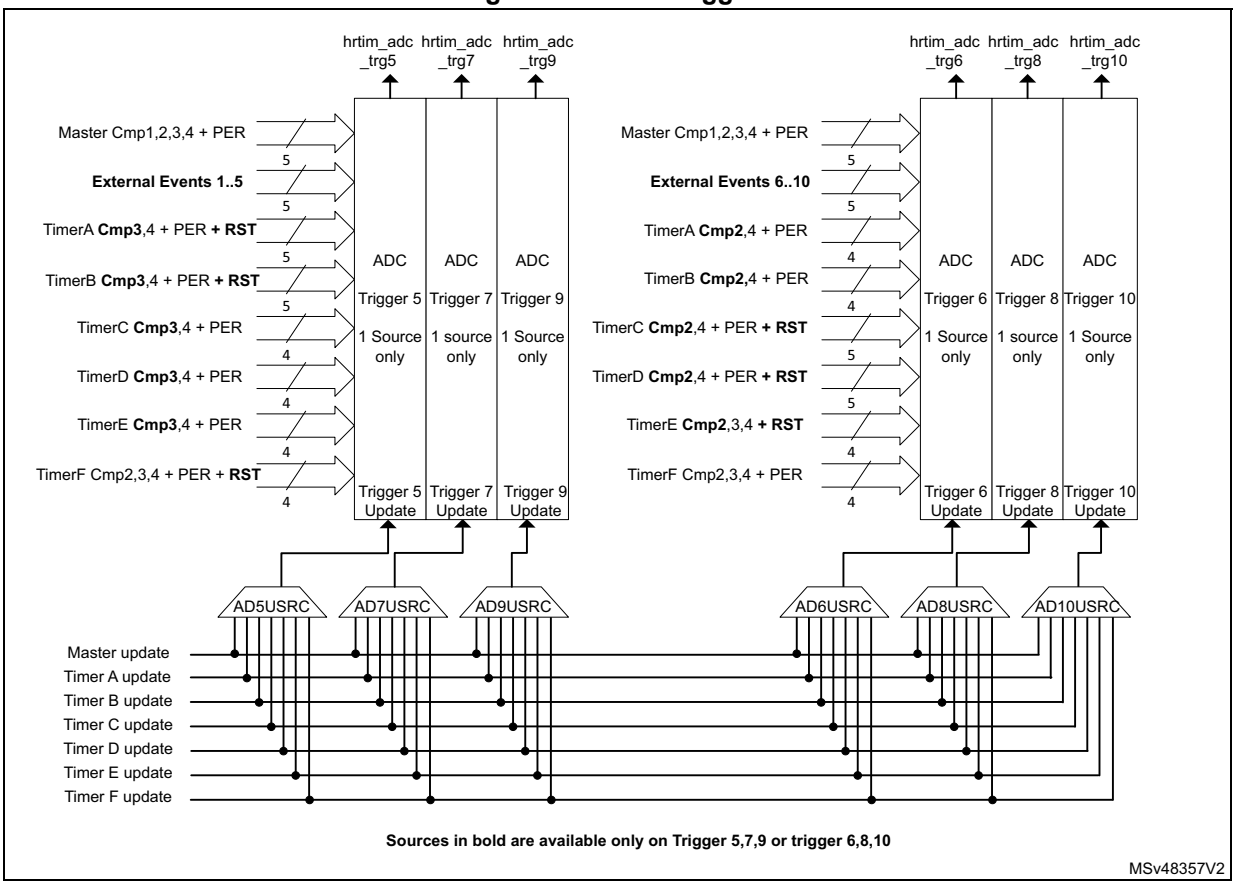
Figure 406. ADC trigger selection overview



The ADC triggers 5 to 10 are configured in the HRTIM_ADCER register, as shown in *Figure 407*. The ADC triggers 5/7/9 and 6/8/10 are using the same source set.

A single source can be selected at once for these triggers (1 out of 32 possible events).

Figure 407. ADC triggers



HRTIM_ADC1R to HRTIM_ADC4R and HRTIM_ADCER registers are preloaded and can be updated synchronously with the timer they are related to. The update sources are defined with ADxUSRC[2:0] bits in the HRTIM_CR1 and HRTIM_ADCUR registers.

For instance, if ADC trigger 1 outputs timer A CMP2 events (HRTIM_ADC1R = 0x0000 0400), HRTIM_ADC1R is typically updated simultaneously with timer A (AD1USRC[2:0] = 001).

When the preload is disabled (PREEN bit reset) in the source timer, the HRTIM_ADCxR registers are not preloaded either: a write access results in an immediate update of the trigger source.

ADC post-scaler

A post-scaling unit allows to reduce the ADC trigger rate as shown in *Figure 408*.

Each ADC trigger rate can be individually adjusted using the ADCxPSC[4:0] bits in the HRTIM_ADCxPS1 and HRTIM_ADCxPS2 registers.

In the center-aligned mode, the ADC trigger rate is also dependent on ADROM[1:0] bit field, programmed in the source timer, as shown in *Figure 409*. The ADROM[1:0] bit field is coding for any event that can trigger the ADC: reset, roll-over (period) and compare event:

- ADROM[1:0] = 00: event generated both during up and down-counting phases
- ADROM[1:0] = 01: event generated during down-counting phases
- ADROM[1:0] = 10: event generated during up-counting phases

The ADC post-scaler programming registers are preloaded and can be updated on-the-fly without stopping the timers.

Figure 408. ADC trigger post-scaling in up-counting mode

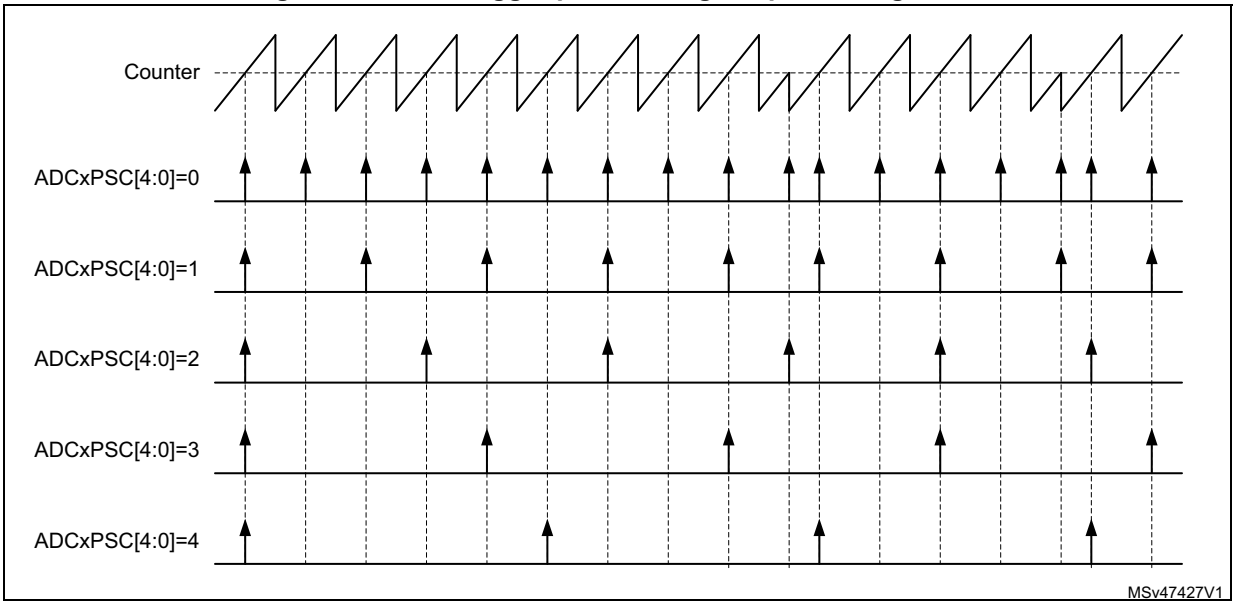
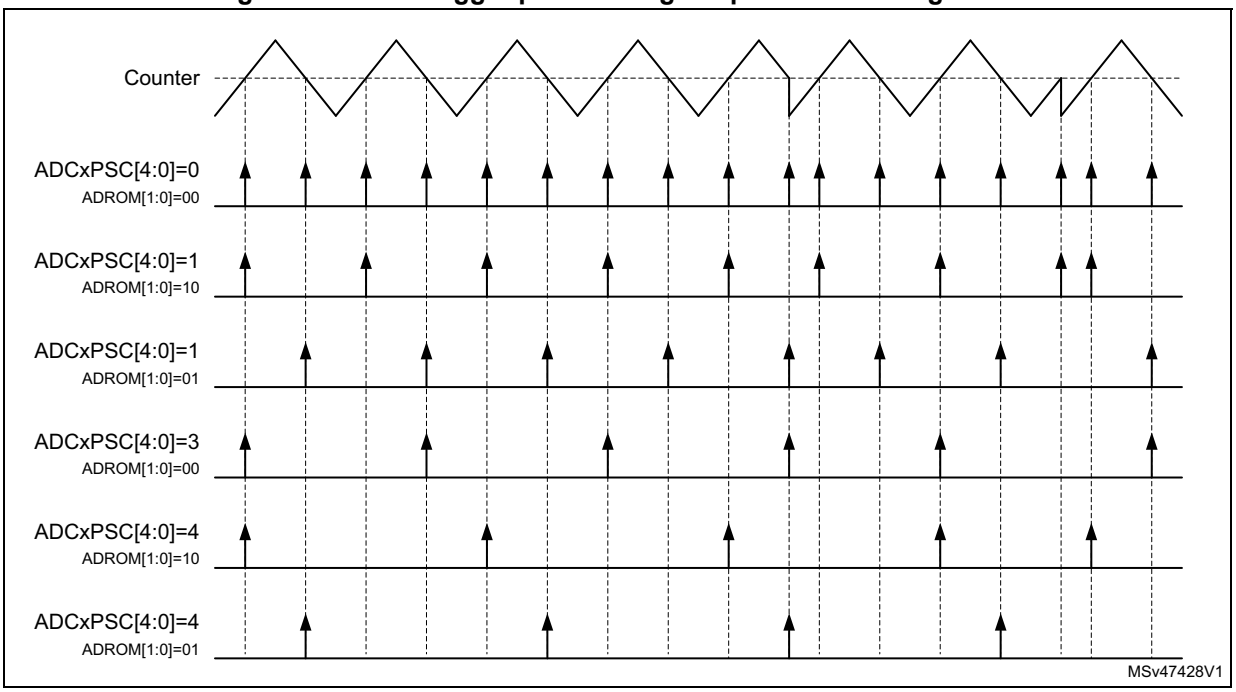


Figure 409. ADC trigger post-scaling in up/down counting mode



34.3.21 DAC triggers

The HRTIM allows to have the embedded DACs updated synchronously with the timer updates.

The update events from the master timer and the timer units can generate DAC update triggers on any of the 3 hrtim_dac_trgx outputs.

Note: Each timer has its own DAC-related control register.

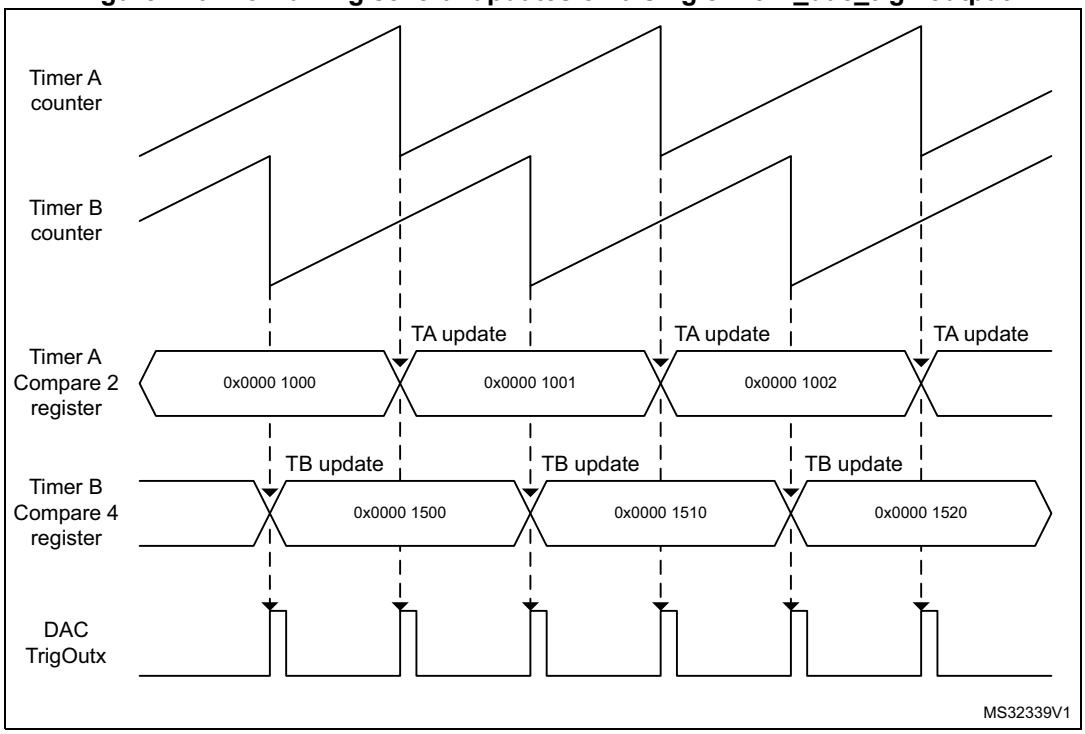
DACSYNC[1:0] bits of the HRTIM_MCR and HRTIM_TIMxCR registers are programmed as follows:

- 00: No update generated
- 01: Update generated on hrtim_dac_trg1
- 10: Update generated on hrtim_dac_trg2
- 11: Update generated on hrtim_dac_trg3

An output pulse of 1 f_{HRTIM} clock periods is generated on the hrtim_dac_trgx output.

When DACSYNC[1:0] bits are enabled in multiple timers, the hrtim_dac_trgx output consists of an OR of all timers' update events. For instance, if DACSYNC = 1 in timer A and in timer B, the update event in timer A is ORed with the update event in timer B to generate a DAC update trigger on the corresponding hrtim_dac_trgx output, as shown in *Figure 410*.

Figure 410. Combining several updates on a single hrtim_dac_trgx output



Refer to HRTIM DAC triggers connections tables of *Chapter 5: Device configuration* for connections to the DACs.

Dual channel DAC trigger

Slope compensation techniques and hysteretic control can be easily implemented using HRTIM built-in features and the DAC sawtooth generator. The principle is to have a DAC generating a decreasing saw-tooth synchronized with the PWM period, or a square wave synchronized with PWM signal.

This mode is enabled with the DCDE bit in the TIMxCR2 register. This bit cannot be changed once the timer is operating (TxEN bit set).

It uses two trigger outputs, as shown in *Figure 411* below:

- The hrtim_dac_reset_trgx generates DAC reset/update events.
- The hrtim_dac_step_trgx generates requests for incremental DAC value changes.

The DCDR bit in the TIMxCR2 register defines when the hrtim_dac_reset_trgx trigger is generated:

- DCDR = 0: the trigger is generated on counter reset or roll-over event.
- DCDR = 1: the trigger is generated on output 1 set event.

Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

The DCDS bit in the TIMxCR2 register defines when the hrtim_dac_step_trgx trigger is generated:

- DCDS = 0: the trigger is generated on compare 2 event.
- DCDS = 1: the trigger is generated on output 1 reset event.

The DCDR and DCDS bits allow the following use cases to be covered:

- Edge-aligned slope compensation (DCDR = DCDS = 0): the DAC's sawtooth starts on PWM period beginning and multiple triggers are generated during the period.
- Center-aligned slope compensation (DCDR = 1 DCDS = 0): the DAC's sawtooth starts on the output set event and multiple triggers are generated during the period.
- Hysteretic controller: the DAC value must be changed twice per period, when the output state changes. 2 triggers are generated per PWM period. In edge-aligned mode (DCDR=0, DCDS =1), the triggers are generated on counter reset or roll-over. In center-aligned mode (DCDR=1, DCDS=1), the triggers are generated when the output is set.

The compare 2 has a particular operating mode when the DCDE is set and the DCDS bit is reset. The active comparison value is automatically updated as soon as a compare match has occurred, so that the trigger can be repeated periodically with a period equal to the CMP2 value, as represented in *Figure 411*.
The dual channel DAC trigger with DCDS bit reset (compare 2 event used) must not be used simultaneously with modes using CMP2 (triple / quad interleaved and triggered-half modes).

Note: The CMP2 value can be changed on-the-fly. The new value is taken into account on the next coming compare match.

When the DCDS bit is reset, the CMP2 value must not be modified by other mechanisms: the interleaved, triggered half and balanced idle modes must be disabled.

Table 406 below gives an example, for generating 6 triggers within a PWM period. It shows that it is necessary to round up the division result to the upper value.

Let’s consider a counter period TIMxPER = 8192. Dividing 8192 by 6 yields 1365.33.

- Round down value: 1365: 7 triggers are generated, the 6th and 7th being very close (respectively for counter = 8190 and 8192).
- Round up value:1366: 6 triggers are generated. The 6th trigger on dac_step_trg (for counter = 8192) is aborted by the counter roll-over from 8192 to 0.

Table 406. DAC dual channel trigger example

-	CMP2 = 1365	dac_trg	dac_step_trg	CMP2 = 1366	dac_trg	dac_step_trg
Counter value	1365	-	1	1366	-	1
	2730	-	2	2732	-	2
	4095	-	3	4098	-	3
	5460	-	4	5464	-	4
	6825	-	5	6830	-	5
	8190	-	6	8192	6	-
	8192	7	-	1366	-	1
	1365	-	1	2732	-	2
	...	-	-	...	-	-

Note: In centered-pattern mode, it is mandatory to have an even number of triggers per switching period, so as to avoid unevenly spaced triggers around counter’s peak value.

Figure 411. DAC triggers for slope compensation

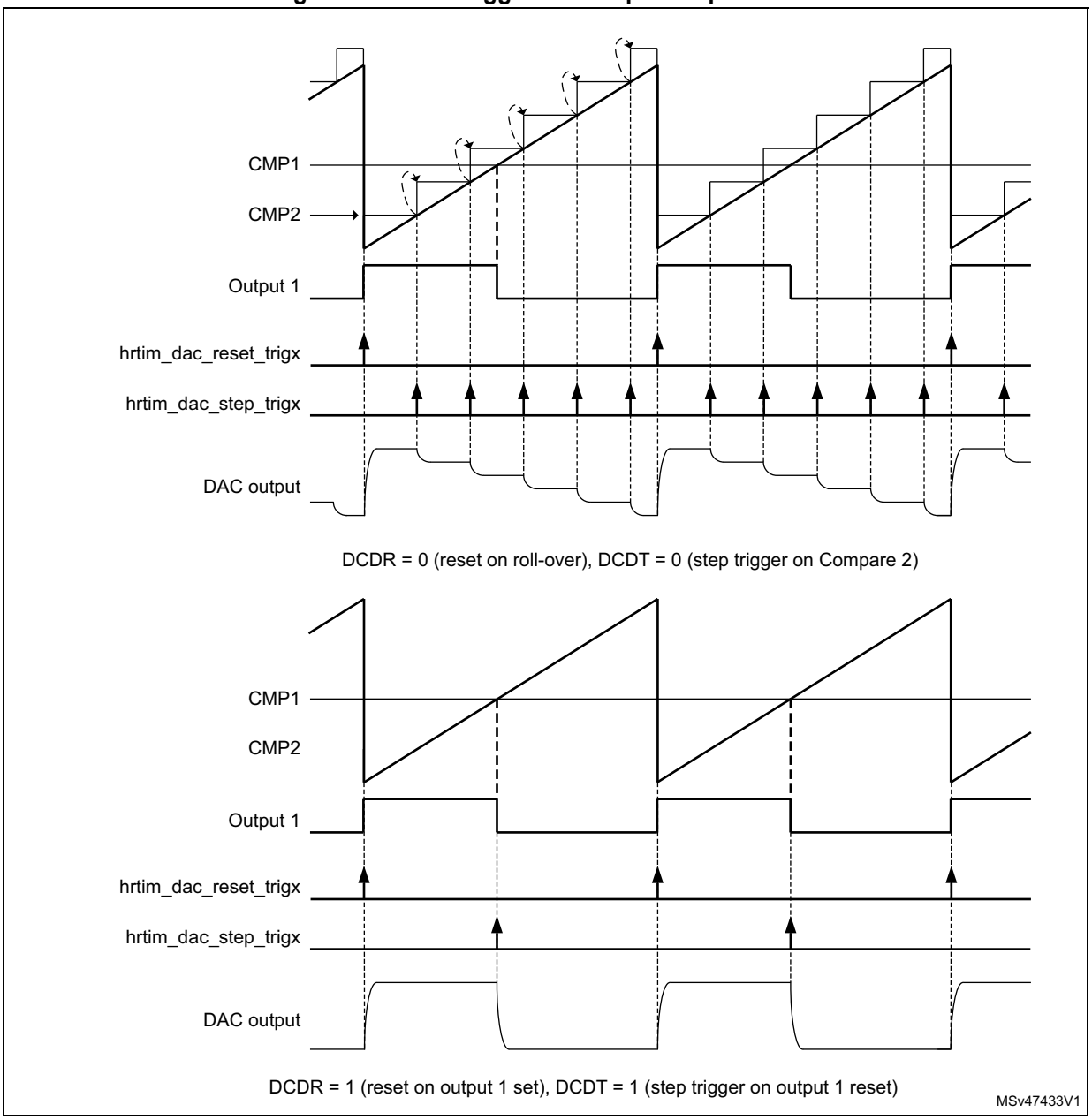
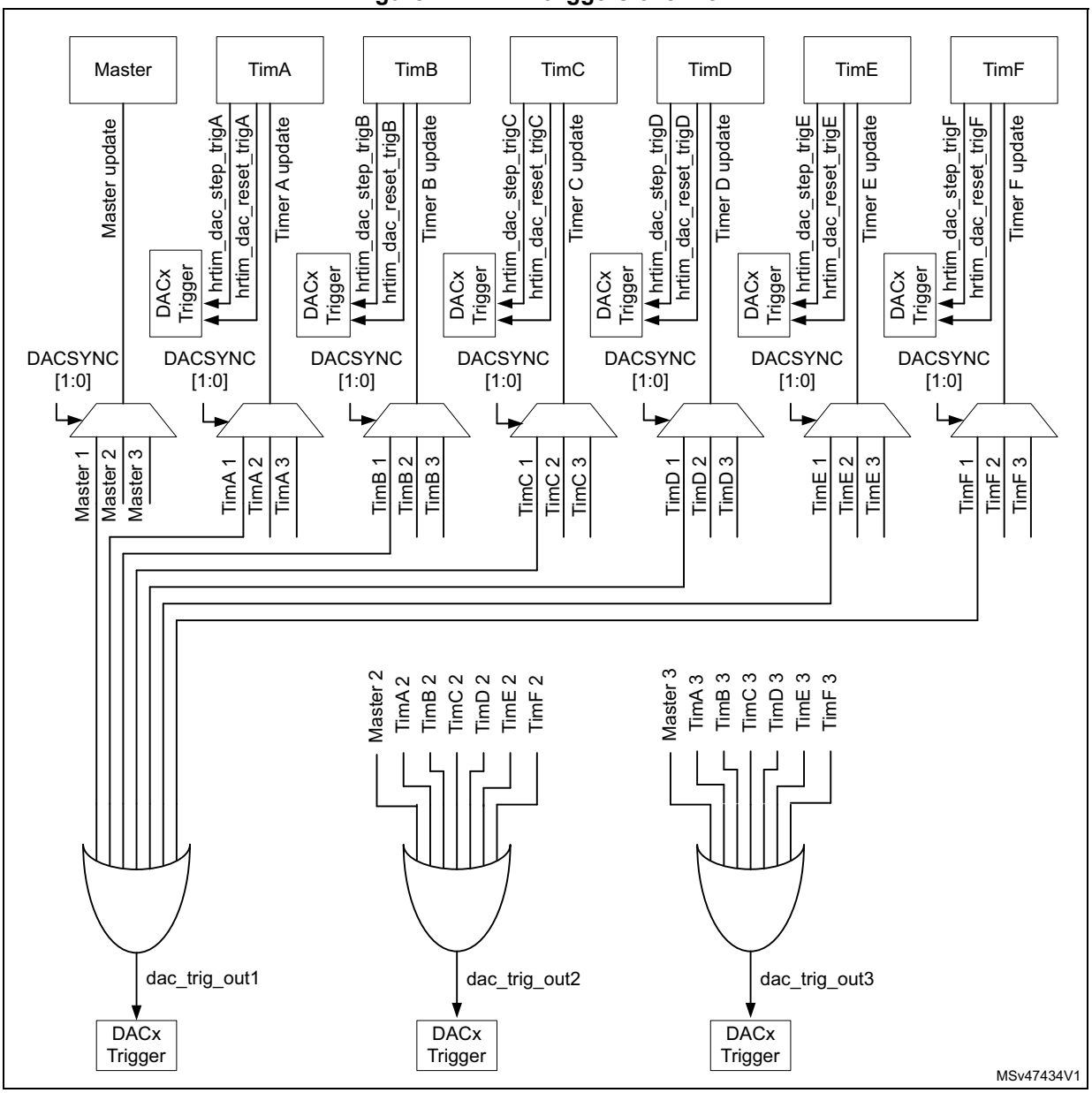


Figure 412 below provides an overview of all the available DAC triggers.

Figure 412. DAC triggers overview



34.3.22 Interrupts

7 interrupts can be generated by the master timer:

- Master timer registers update
- Synchronization event received
- Master timer repetition event
- Master compare 1 to 4 event

14 interrupts can be generated by each timing unit:

- Delayed protection triggered
- Counter reset or roll-over event
- Output 1 and output 2 reset (transition active to inactive)
- Output 1 and output 2 set (transition inactive to active)
- Capture 1 and 2 events
- Timing unit registers update
- Repetition event
- Compare 1 to 4 event

8 global interrupts are generated for the whole HRTIM:

- System fault and fault 1 to 6 (regardless of the timing unit attribution)
- DLL calibration done
- Burst mode period completed

The interrupt requests are grouped in 8 vectors as follows:

- hrtim_it1: master timer interrupts (master update, sync Input, repetition, MCMP1..4) and global interrupt except faults (burst mode period and DLL ready interrupts)
- hrtim_it2: TIMA interrupts
- hrtim_it3: TIMB interrupts
- hrtim_it4: TIMC interrupts
- hrtim_it5: TIMD interrupts
- hrtim_it6: TIME interrupts
- hrtim_it7: TIMF interrupts
- hrtim_it8: Dedicated vector all fault interrupts to allow high-priority interrupt handling

Table 407 is a summary of the interrupt requests, their mapping and associated control, and status bits.

Table 407. HRTIM interrupt summary

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
hrtim_it1	Burst mode period completed	BMPER	BMPERIE	BMPERC
	DLL calibration done	DLLRDY	DLLRDYIE	DLLRDYC
	Master timer registers update	MUPD	MUPDIE	MUPDC
	Synchronization event received	SYNC	SYNCIE	SYNCC
	Master timer repetition event	MREP	MREPIE	MREPC
	Master compare 1 to 4 event	MCMP1	MCMP1IE	MCP1C
		MCMP2	MCMP2IE	MCP2C
		MCMP3	MCMP3IE	MCP3C
		MCMP4	MCMP4IE	MCP4C

Table 407. HRTIM interrupt summary (continued)

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
hrtim_it2 hrtim_it3 hrtim_it4 hrtim_it5 hrtim_it6 hrtim_it7	Delayed protection triggered	DLYPRT	DLYPRTIE	DLYPRTC
	Counter reset or roll-over event	RST	RSTIE	RSTC
	Output 1 and output 2 reset (transition active to inactive)	RSTx1	RSTx1IE	RSTx1C
		RSTx2	RSTx2IE	RSTx2C
	Output 1 and output 2 set (transition inactive to active)	SETx1	SETx1IE	SETx1C
		SETx2	SETx2IE	SETx2C
	Capture 1 and 2 events	CPT1	CPT1IE	CPT1C
		CPT2	CPT2IE	CPT2C
	Timing unit registers update	UPD	UPDIE	UPDC
	Repetition event	REP	REPIE	REPC
	Compare 1 to 4 event	CMP1	CMP1IE	CMP1C
		CMP2	CMP2IE	CMP2C
		CMP3	CMP3IE	CMP3C
		CMP4	CMP4IE	CMP4C
hrtim_it8	System fault	SYSFLT	SYSFLTIE	SYSFLTC
	Fault 1 to 6	FLT1	FLT1IE	FLT1C
		FLT2	FLT2IE	FLT2C
		FLT3	FLT3IE	FLT3C
		FLT4	FLT4IE	FLT4C
		FLT5	FLT5IE	FLT5C
		FLT6	FLT6IE	FLT6C

34.3.23 DMA

Most of the events able to generate an interrupt can also generate a DMA request, even both simultaneously. Each timer (master, TIMA...F) has its own DMA enable register.

The individual DMA requests are ORed into 7 channels as follows:

- 1 channel for the master timer
- 1 channel per timing unit (TIMA...F)

Note: Before disabling a DMA channel (DMA enable bit reset in TIMxDIER), it is necessary to disable first the DMA controller.

Table 408 is a summary of the events with their associated DMA enable bits.

Table 408. HRTIM DMA request summary

DMA Channel	Event	DMA capable	DMA enable bit
hrtim_dma1 (master timer)	Burst mode period completed	No	N/A
	DLL calibration done	No	N/A
	Master timer registers update	Yes	MUPDDE
	Synchronization event received	Yes	SYNCDE
	Master timer repetition event	Yes	MREPDE
	Master compare 1 to 4 event	Yes	MCMP1DE
		Yes	MCMP2DE
		Yes	MCMP3DE
		Yes	MCMP4DE
hrtim_dma2 (timer A) hrtim_dma3 (timer B) hrtim_dma4 (timer C) hrtim_dma5 (timer D) hrtim_dma6 (timer E) hrtim_dma7 (timer F)	Delayed protection triggered	Yes	DLYPRTDE
	Counter reset or roll-over event	Yes	RSTDE
	Output 1 and output 2 reset (transition active to inactive)	Yes	RSTx1DE
		Yes	RSTx2DE
	Output 1 and output 2 set (transition inactive to active)	Yes	SETx1DE
		Yes	SETx2DE
	Capture 1 and 2 events	Yes	CPT1DE
		Yes	CPT2DE
	Timing unit registers update	Yes	UPDDE
	Repetition event	Yes	REPDE
	Compare 1 to 4 event	Yes	CMP1DE
		Yes	CMP2DE
		Yes	CMP3DE
		Yes	CMP4DE
N/A	System fault	No	N/A
	Fault 1 to 6	No	N/A
	Burst mode period completed	No	N/A
	DLL calibration done	No	N/A

Burst DMA transfers

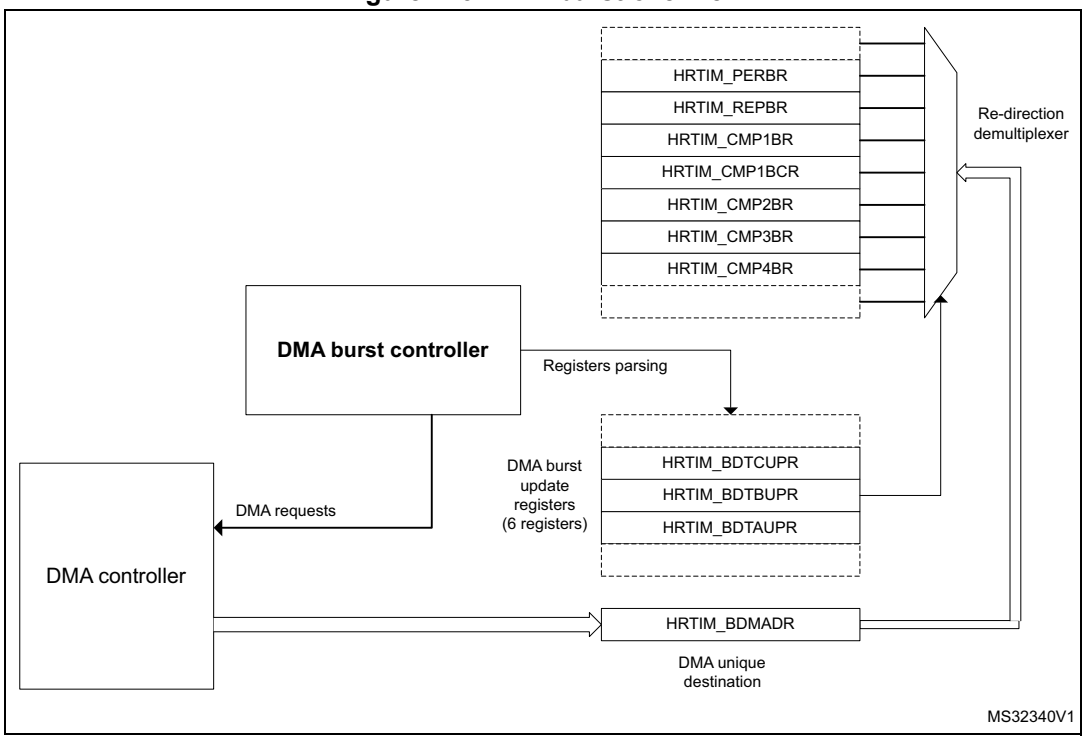
In addition to the standard DMA requests, the HRTIM features a DMA burst controller to have multiple registers updated with a single DMA request. This allows to:

- Update multiple data registers with one DMA channel only,
- Reprogram dynamically one or several timing units, for converters using multiple timer outputs.

The burst DMA feature is only available for one DMA channel, but any of the 6 channels can be selected for burst DMA transfers.

The principle is to program which registers are to be written by DMA. The master timer and TIMA..E have the burst DMA update register, where most of their control and data registers are associated with a selection bit: HRTIM_BDMUPR, HRTIM_BDTAUPR to HRTIM_BDTEUPR (this is applicable only for registers with write accesses). A redirection mechanism allows to forward the DMA write accesses to the HRTIM registers automatically, as shown in *Figure 413*.

Figure 413. DMA burst overview



When the DMA trigger occurs, the HRTIM generates multiple 32-bit DMA requests and parses the update register. If the control bit is set, the write access is redirected to the associated register. If the bit is reset, the register update is skipped and the register parsing is resumed until a new bit set is detected, to trigger a new request. Once the 6 update registers (HRTIM_BDMUPR, 5x HRTIM_BDTxUPR) are parsed, the burst is completed and the system is ready for another DMA trigger (refer to the flowchart in *Figure 414*).

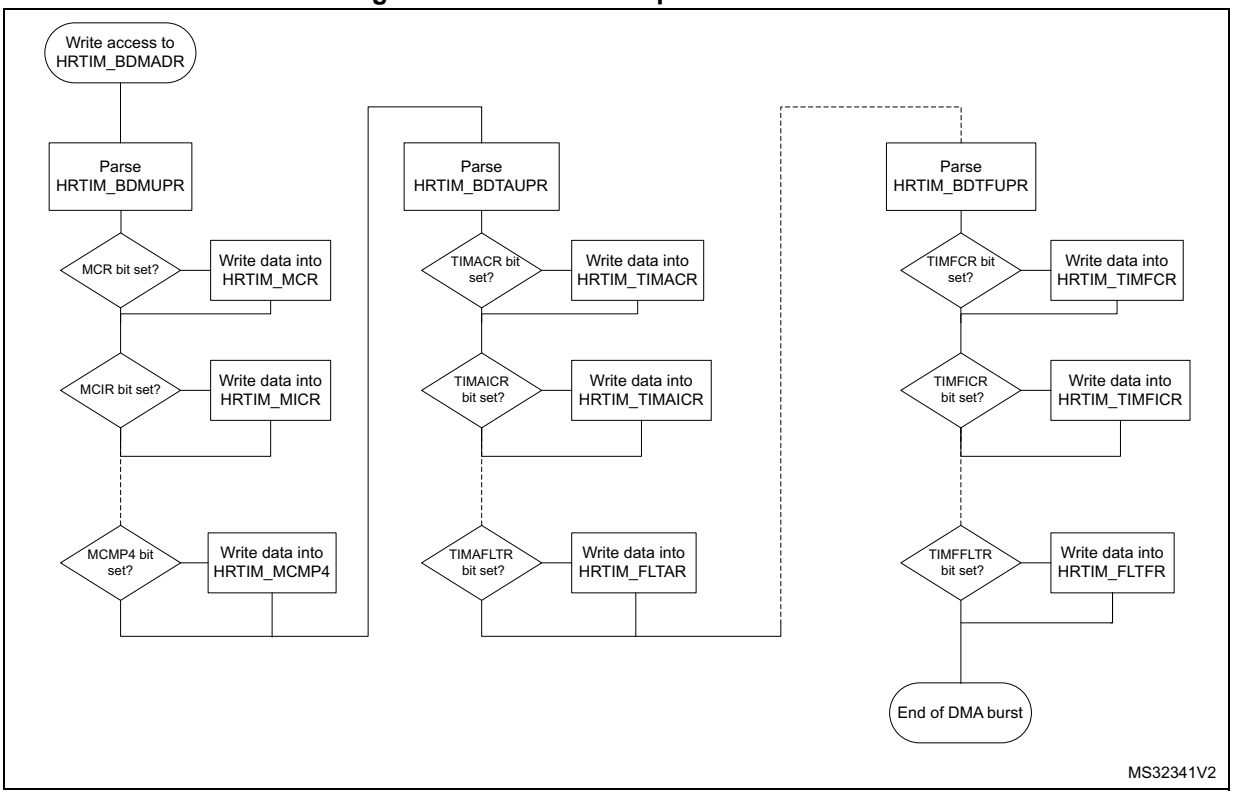
Note: Any trigger occurring while the burst is on-going is discarded, except if it occurs during the very last data transfer.

The burst DMA mode is permanently enabled (there is no enable bit). A burst DMA operation is started by the first write access into the HRTIM_BDMADR register.

It is only necessary to have the DMA controller pointing to the HRTIM_BDMADR register as the destination, in the memory, to the peripheral configuration with the peripheral increment mode disabled (the HRTIM handles internally the data re-routing to the final destination register).

To re-initialize the burst DMA mode if it was interrupted during a transaction, it is necessary to write at least to one of the 6 update registers.

Figure 414. Burst DMA operation flowchart



Several options are available once the DMA burst is completed, depending on the register update strategy.

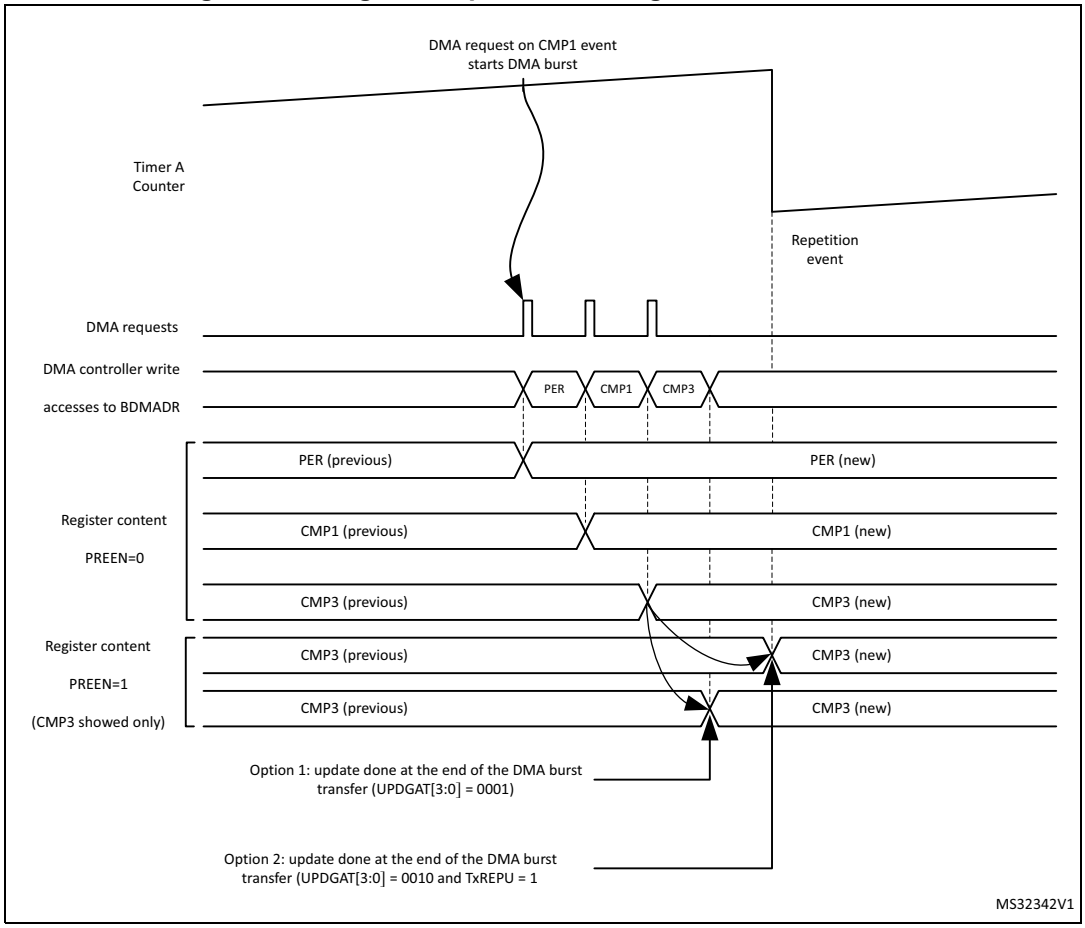
If the PREEN bit is reset (preload disabled), the value written by the DMA is immediately transferred into the active register and the registers are updated sequentially, following the DMA transaction pace.

When the preload is enabled (PREEN bit set), there are 3 use cases:

1. The update is done independently from DMA burst transfers (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 00 in HRTIM_MCR). In this case, and if it is necessary to have all transferred data taken into account simultaneously, the user must check that the DMA burst is completed before the update event takes place. On the contrary, if the update event happens while the DMA transfer is on-going, only part of the registers is loaded and the complete register update requires 2 consecutive update events.
2. The update is done when the DMA burst transfer is completed (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 01 in HRTIM_MCR). This mode guarantees that all new register values are transferred simultaneously. This is done independently from the counter value and can be combined with regular update events, if necessary (for instance, an update on a counter reset when TxRSTU is set).
3. The update is done on the update event following the DMA burst transfer completion (UPDGAT[3:0] = 0010 in HRTIM_TIMxCR and BRSTDMA[1:0] = 10 in HRTIM_MCR). This mode guarantees both a coherent update of all transferred data and the synchronization with regular update events, with the timer counter. In this case, if a regular update request occurs while the transfer is on-going, it is discarded and the effective update happens on the next coming update request.

The chronogram in *Figure 415* presents the active register content for 3 cases: PREEN=0, UPDGAT[3:0] = 0001 and UPDGAT[3:0] = 0001 (when PREEN = 1).

Figure 415. Registers update following DMA burst transfer



34.3.24 HRTIM initialization

This section describes the recommended HRTIM initialization procedure, including other related MCU peripherals.

The HRTIM clock source must be enabled in the reset and clock control unit (RCC), while respecting the $_{fHRTIM}$ range for the DLL lock.

The DLL calibration must be started by setting CAL bit in HRTIM_DLLCR register.

The HRTIM master and timing units can be started only once the high-resolution unit is ready. This is indicated by the DLLRDY flag set. The DLLRDY flag can be polled before resuming the initialization or the calibration can run in background while other registers of the HRTIM or other MCU peripherals are initialized. In this case, the DLLRDY flag must be checked before starting the counters (an end-of-calibration interrupt can be issued if necessary, enabled with DLLRDYIE flag in HRTIM_IER). Once the DLL calibration is done, CALEN bit must be set to have it done periodically and compensate for potential voltage and temperature drifts. The calibration periodicity is defined using the CALRTE[1:0] bit field in the HRTIM_DLLCR register.

The HRTIM control registers can be initialized as per the power converter topology and the timing units use case. All inputs have to be configured (source, polarity, edge-sensitivity).

The HRTIM outputs must be set up eventually, with the following sequence:

- The polarity must be defined using POLx bits in HRTIM_OUTxR.
- The FAULT and IDLE states must be configured using FAULTx[1:0] and IDLESx bits in HRTIM_OUTxR.

The HRTIM outputs are ready to be connected to the MCU I/Os. In the GPIO controller, the selected HRTIM I/Os have to be configured as per the alternate function mapping table in the SR5E1x pinout Microsoft® Excel® file attached to the IO definition document.

From this point on, the HRTIM controls the outputs, which are in the IDLE state.

The outputs are configured in RUN mode by setting TxyOEN bits in the HRTIM_OENR register. The 2 outputs are in the inactive state until the first valid set/reset event in RUN mode. Any output set/reset event (except software requests using SST, SRT) is ignored as long as TxCEN bit is reset, as well as burst mode requests (IDLEM bit value is ignored). Similarly, any counter reset request coming from the burst mode controller is ignored (if TxBM bit is set).

Note: When the deadtime insertion is enabled (DTEN bit set), it is necessary to force the output state by software, using SST and RST bits, to have the outputs in a complementary state as soon as the RUN mode is entered.

The HRTIM operation can eventually be started by setting TxCEN or MCEN bits in HRTIM_MCR.

If the HRTIM peripheral is reset with the Reset and Clock Controller, the output control is released to the GPIO controller and the outputs are tri-stated.

34.3.25 Debug

When a microcontroller enters the debug mode (Cortex®-M7 core halted), the TIMx counter either continues to work normally or stops, depending on HRTIMx_FRZ configuration bit in DBG module:

- HRTIMx_FRZ = 0: no behavior change, the HRTIM continues to operate.
- HRTIMx_FRZ = 1: all HRTIM timers, including the master, are stopped. The outputs in RUN mode enter the FAULT state if FAULTx[1:0] = 01,10,11, or keep their current state if FAULTx[1:0] = 00. The outputs in idle state are maintained in this state. This is permanently maintained even if the MCU exits the halt mode. This allows to maintain a safe state during the execution stepping. The outputs can be enabled again by settings TxyOEN bit (requires the use of the debugger).

Timer behavior during MCU halt when HRTIMx_FRZ = 1

The set/reset crossbar, the dead-time and push-pull unit, the idle/balanced fault detection and all the logic driving the normal output in RUN mode are not affected by debug. The output keeps on toggling internally, so as to retrieve regular signals of the outputs when TxyOEN is set again (during or after the MCU halt). Associated triggers and filters are also following internal waveforms when the outputs are disabled.

FAULT inputs and events (any source) are enabled during the MCU halt.

Fault status bits can be set and TxyOEN bits reset during the MCU halt if a fault occurs at that time (TxyOEN and TxyODS are not affected by HRTIMx_FRZ bit state).

Synchronization, counter reset, start and reset-start events are discarded in debug mode, as well as capture events. This is to keep all related registers stable as long as the MCU is halted.

The counter stops counting when a breakpoint is reached. However, the counter enable signal is not reset; consequently no start event is emitted when exiting from debug. All counter reset and capture triggers are disabled, as well as external events (ignored as long as the MCU is halted). The outputs SET and RST flags are frozen, except in case of forced software set/reset. A level-sensitive event is masked during the debug but is active again as soon as the debug is exited. For edge-sensitive events, if the signal is maintained active during the MCU halt, a new edge is not generated when exiting from debug.

The update events are discarded. This prevents any update trigger on hrtim_upd_en[3:1] inputs. DMA triggers are disabled. The burst mode circuit is frozen: the triggers are ignored and the burst mode counter stopped.

DLL calibration is not blocked while the MCU is halted (the DLLRDY flag can be set).

34.4 Application use cases

34.4.1 Buck converter

Buck converters are of common use as step-down converters. The HRTIM can control up to 12 buck converters with 7 independent switching frequencies.

The converter usually operates at a fixed frequency and the Vin/Vout ratio depends on the duty cycle D applied to the power switch:.

$$V_{out} = D \times V_{in}$$

The topology is given in *Figure 416* with the connection to the ADC for voltage reading.

Figure 416. Buck converter topology

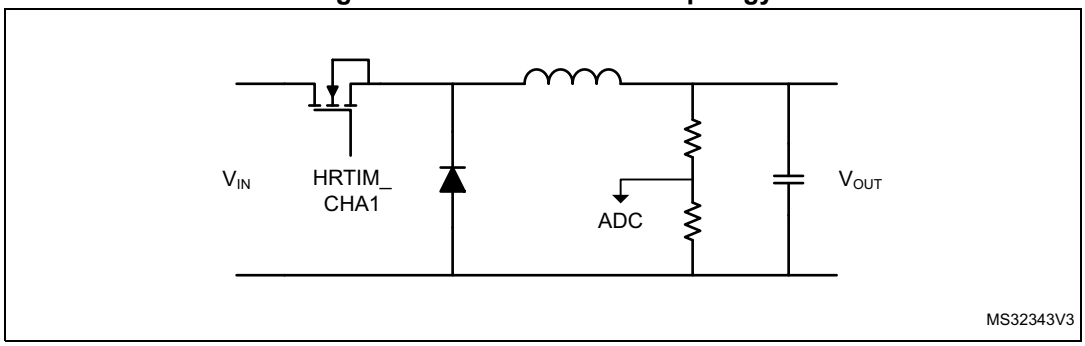
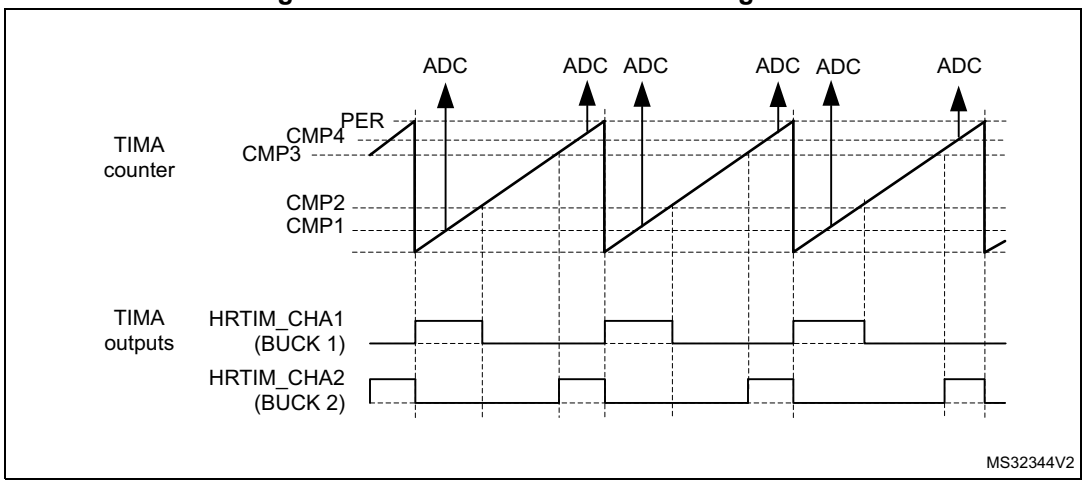


Figure 417 presents the management of two converters with identical frequency PWM signals. The outputs are defined as follows:

- HRTIM_CHA1 set on period, reset on CMP1.
- HRTIM_CHA2 set on CMP3, reset on PER.

The ADC is triggered twice per period, precisely in the middle of the ON time, using CMP2 and CMP4 events.

Figure 417. Dual Buck converter management

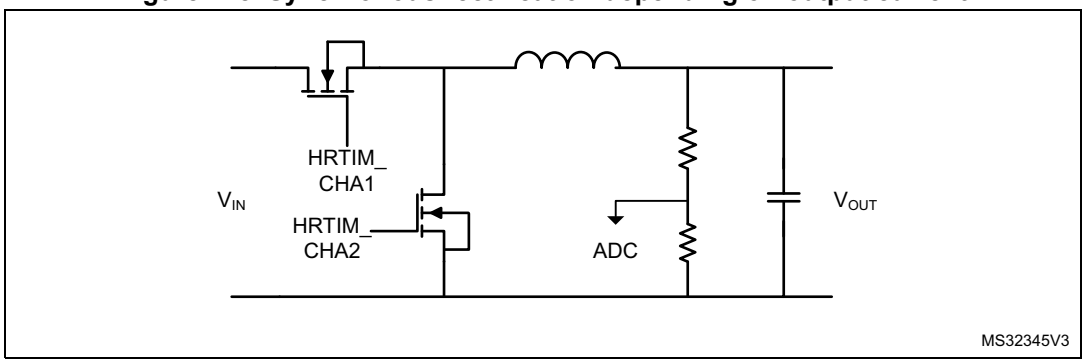


Timers A..E provide either 12 buck converters coupled by pairs (both with identical switching frequencies) or 7 completely independent converters (each of them having a different switching frequency), using the master timer as the 7th time base.

34.4.2 Buck converter with synchronous rectification

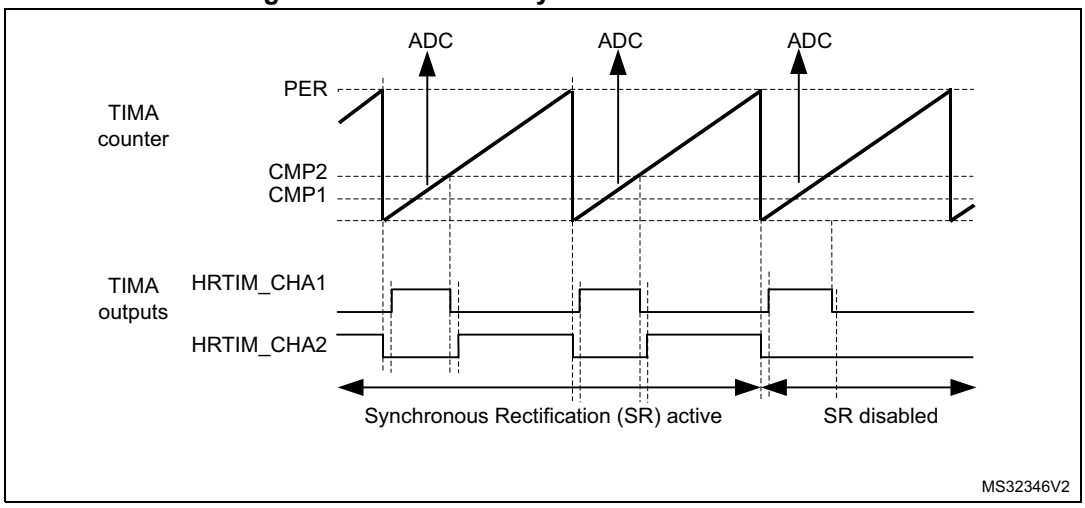
Synchronous rectification allows to minimize losses in buck converters, by means of a FET replacing the freewheeling diode. Synchronous rectification can be turned on or off on the fly depending on the output current level, as shown in *Figure 418*.

Figure 418. Synchronous rectification depending on output current



The main difference versus a single-switch buck converter is the addition of a deadtime for an almost complementary waveform generation on HRTIM_CHA2, based on the reference waveform on HRTIM_CHA1 (refer to *Figure 419*).

Figure 419. Buck with synchronous rectification



34.4.3 Multiphase converters

Multiphase techniques can be applied to multiple power conversion topologies (buck, flyback). Their main benefits are:

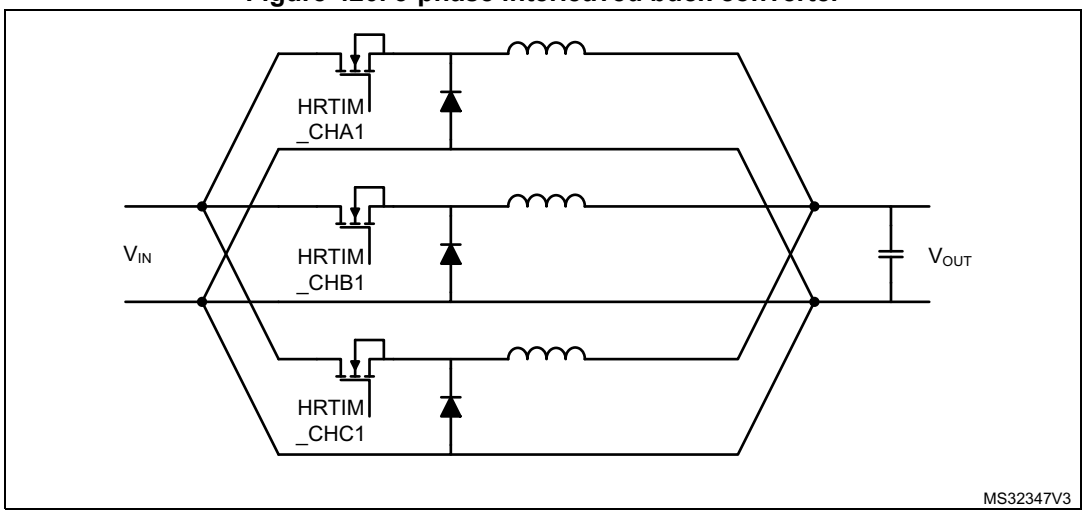
- Reduction of the current ripple on the input and output capacitors
- Reduced EMI
- Higher efficiency at light load by dynamically changing the number of phases (phase shedding)

The HRTIM manages multiple converters. The number of converters that can be controlled depends on the topologies and resources used (including the ADC triggers):

- 5 buck converters with synchronous rectification (SR), using the master timer and the 5 timers
- 4 buck converters (without SR), using the master timer and 2 timers

Figure 420 presents the topology of a 3-phase interleaved buck converter.

Figure 420. 3-phase interleaved buck converter



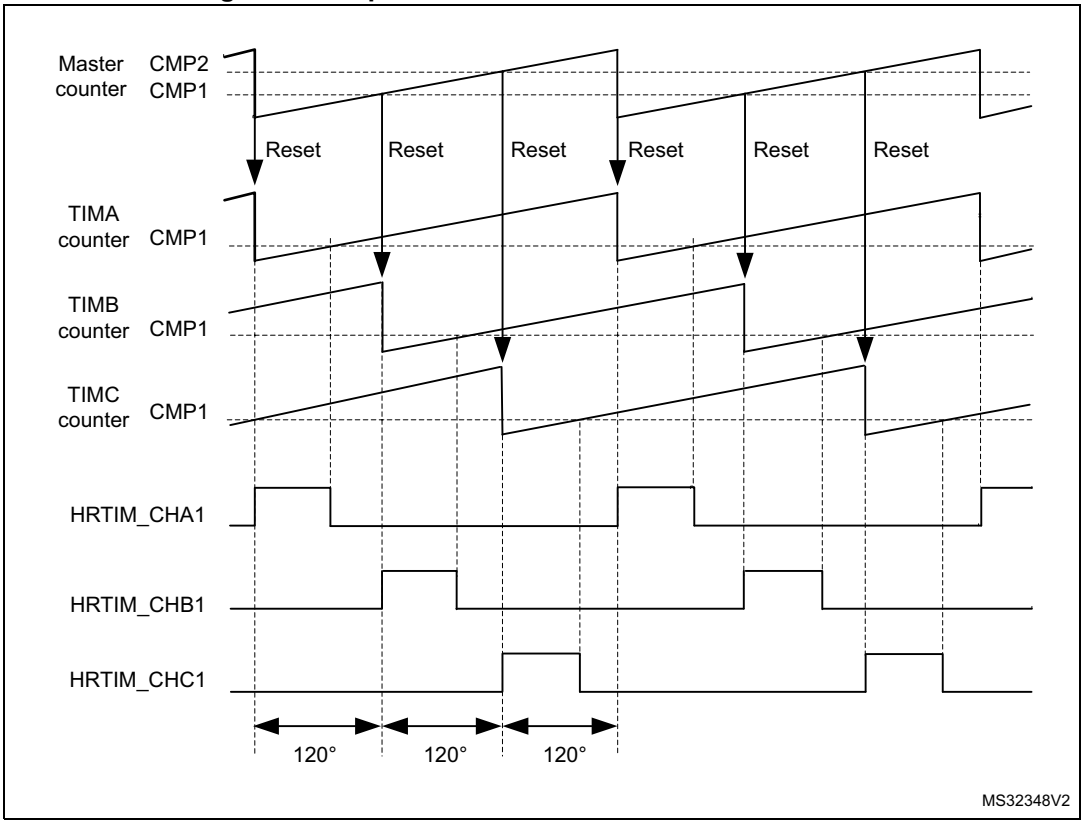
The master timer is responsible for the phase management: it defines the phase relationship between the converters by resetting the timers periodically. The phase-shift is 360° divided by the number of phases, 120° in the given example.

The duty cycle is then programmed into each of the timers. The outputs are defined as follows:

- HRTIM_CHA1 set on master timer period, reset on TACMP1
- HRTIM_CHB1 set on master timer MCMP1, reset on TBCMP1
- HRTIM_CHC1 set on master timer MCMP2, reset on TCCMP1

The ADC trigger can be generated on TxCMP2 compare event. Since all ADC trigger sources are phase-shifted because of the converter topology, it is possible to have all of them combined into a single ADC trigger to save ADC resources (for instance 1 ADC regular channel for the full multi-phase converter).

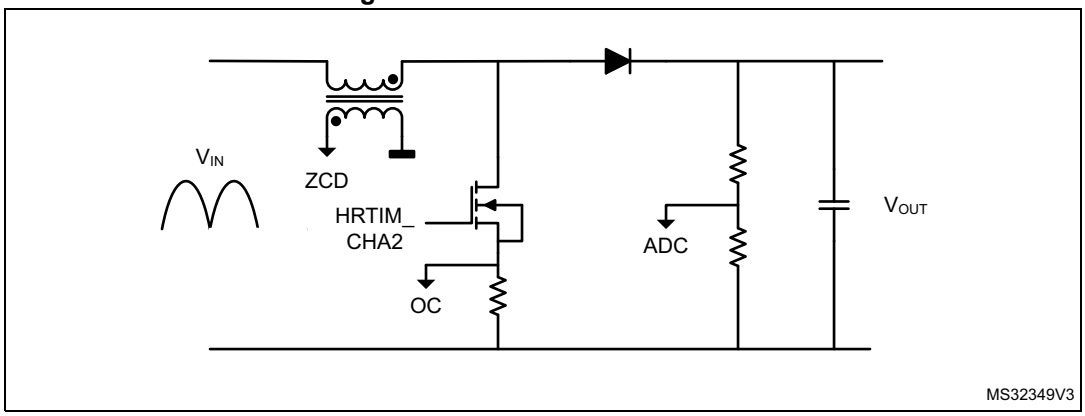
Figure 421. 3-phase interleaved buck converter control



34.4.4 Transition mode power factor correction

The basic operating principle is to build up current into an inductor during a fixed Ton time. This current then decays during the T_{off} time, and the period is re-started when it becomes null. This is detected using a Zero Crossing Detection circuitry (ZCD), as shown in *Figure 422*. With a constant T_{on} time, the peak current value in the inductor is proportional to the rectified AC input voltage, which provides the power factor correction.

Figure 422. Transition mode PFC



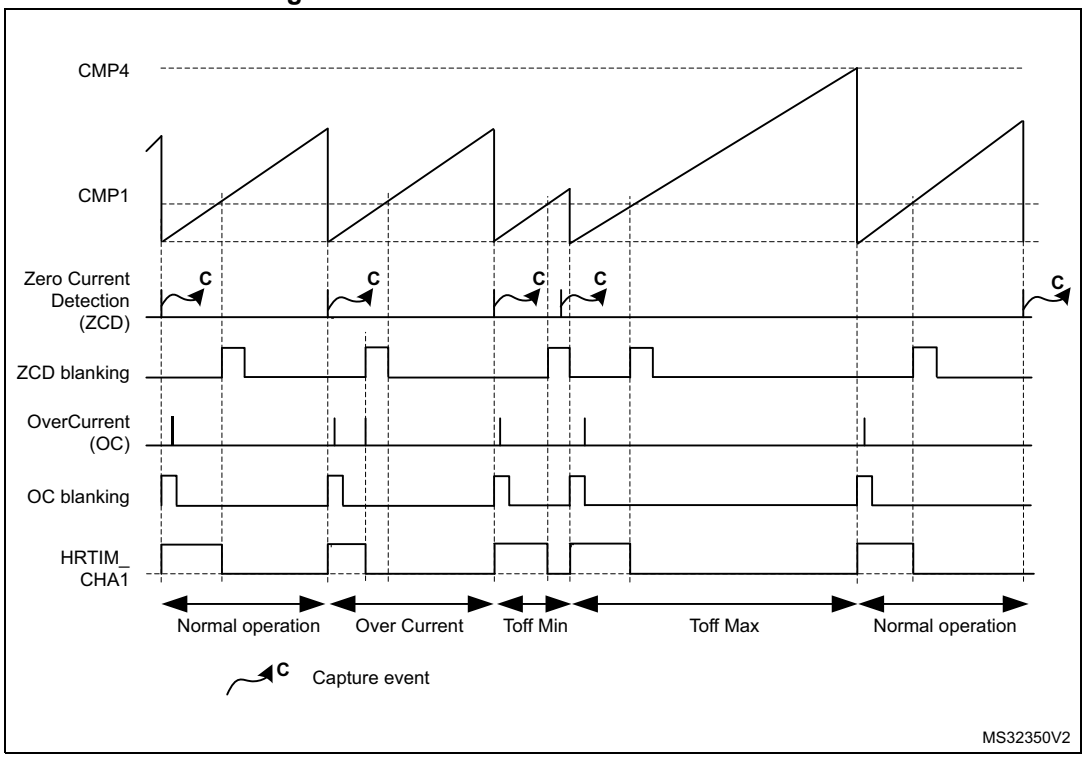
This converter operates with a constant Ton time and a variable frequency due the T_{off} time variation (depending on the input voltage). It must also include some features to operate when no zero-crossing is detected, or to limit the Ton time in case of over-current (OC). The OC feedback is usually conditioned with the built-in comparator and routed onto an external event input.

Figure 423 presents the waveform during the various operating modes, with the following defined parameters:

- T_{on} Min: masks spurious overcurrent (freewheeling diode recovery current), represented as OC blanking.
- T_{on} Max: practically, the converter set-point. It is defined by CMP1.
- T_{off} Min: limits the frequency when the current limit is close to zero (demagnetization is very fast). It is defined with CMP2.
- T_{off} Max: prevents the system to be stuck if no ZCD occurs. It is defined with CMP4 in auto-delayed mode.

Both T_{off} values are auto-delayed since the value must be relative to the output falling edge.

Figure 423. Transition mode PFC waveforms



34.5 HRTIM registers

34.5.1 HRTIM register memory map

Table 409. HRTIM register memory map

Offset	Register name
0x000	*HRTIM master timer control register (HRTIM_MCR)*
0x004	*HRTIM master timer interrupt status register (HRTIM_MISR)*
0x008	*HRTIM master timer interrupt clear register (HRTIM_MICR)*
0x00C	*HRTIM master timer DMA interrupt enable register (HRTIM_MDIER)*
0x010	*HRTIM master timer counter register (HRTIM_MCNTR)*
0x014	*HRTIM master timer period register (HRTIM_MPER)*
0x018	*HRTIM master timer repetition register (HRTIM_MREP)*
0x01C	*HRTIM master timer compare 1 register (HRTIM_MCMP1R)*
0x024	*HRTIM master timer compare 2 register (HRTIM_MCMP2R)*
0x028	*HRTIM master timer compare 3 register (HRTIM_MCMP3R)*
0x02C	*HRTIM master timer compare 4 register (HRTIM_MCMP4R)*
Block A: 0x080 Block B: 0x100 Block C: 0x180 Block D: 0x200 Block E: 0x280 Block F: 0x300	*HRTIM timer x control register (HRTIM_TIMxCR) (x = A to F)*
Block A: 0x084 Block B: 0x104 Block C: 0x184 Block D: 0x204 Block E: 0x284 Block F: 0x304	*HRTIM timer x interrupt status register (HRTIM_TIMxISR) (x = A to F)*
Block A: 0x088 Block B: 0x108 Block C: 0x188 Block D: 0x208 Block E: 0x288 Block F: 0x308	*HRTIM timer x interrupt clear register (HRTIM_TIMxICR) (x = A to F)*
Block A: 0x08C Block B: 0x10C Block C: 0x18C Block D: 0x20C Block E: 0x28C Block F: 0x30C	*HRTIM timer x DMA interrupt enable register (HRTIM_TIMxDIER) (x = A to F)*

Table 409. HRTIM register memory map (continued)

Offset	Register name
Block A: 0x090 Block B: 0x110 Block C: 0x190 Block D: 0x210 Block E: 0x290 Block F: 0x310	*HRTIM timer x counter register (HRTIM_CNTxR) (x = A to F)*
Block A: 0x094 Block B: 0x114 Block C: 0x194 Block D: 0x214 Block E: 0x294 Block F: 0x314	*HRTIM timer x period register (HRTIM_PERxR) (x = A to F)*
Block A: 0x098 Block B: 0x118 Block C: 0x198 Block D: 0x218 Block E: 0x298 Block F: 0x318	*HRTIM timer x repetition register (HRTIM_REPxR) (x = A to F)*
Block A: 0x09C Block B: 0x11C Block C: 0x19C Block D: 0x21C Block E: 0x29C Block F: 0x31C	*HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F)*
Block A: 0x0A0 Block B: 0x120 Block C: 0x1A0 Block D: 0x220 Block E: 0x2A0 Block F: 0x320	*HRTIM timer x compare 1 compound register (HRTIM_CMP1CxR) (x = A to F)*
Block A: 0x0A4 Block B: 0x124 Block C: 0x1A4 Block D: 0x224 Block E: 0x2A4 Block F: 0x324	*HRTIM timer x compare 2 register (HRTIM_CMP2xR) (x = A to F)*
Block A: 0x0A8 Block B: 0x128 Block C: 0x1A8 Block D: 0x228 Block E: 0x2A8 Block F: 0x328	*HRTIM timer x compare 3 register (HRTIM_CMP3xR) (x = A to F)*

Table 409. HRTIM register memory map (continued)

Offset	Register name
Block A: 0x0AC Block B: 0x12C Block C: 0x1AC Block D: 0x22C Block E: 0x2AC Block F: 0x32C	*HRTIM timer x compare 4 register (HRTIM_CMP4xR) (x = A to F)*
Block A: 0x0B0 Block B: 0x130 Block C: 0x1B0 Block D: 0x230 Block E: 0x2B0 Block F: 0x330	*HRTIM timer x capture 1 register (HRTIM_CPT1xR) (x = A to F)*
Block A: 0x0B4 Block B: 0x134 Block C: 0x1B4 Block D: 0x234 Block E: 0x2B4 Block F: 0x334	*HRTIM timer x capture 2 register (HRTIM_CPT2xR) (x = A to F)*
Block A: 0x0B8 Block B: 0x138 Block C: 0x1B8 Block D: 0x238 Block E: 0x2B8 Block F: 0x338	*HRTIM timer x deadtime register (HRTIM_DTxR) (x = A to F)*
Block A: 0x0BC Block B: 0x13C Block C: 0x1BC Block D: 0x23C Block E: 0x2BC Block F: 0x33C	*HRTIM timer x output 1 set register (HRTIM_SETx1R) (x = A to F)*
Block A: 0x0C0 Block B: 0x140 Block C: 0x1C0 Block D: 0x240 Block E: 0x2C0 Block F: 0x340	*HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F)*
Block A: 0x0C4 Block B: 0x144 Block C: 0x1C4 Block D: 0x244 Block E: 0x2C4 Block F: 0x344	*HRTIM timer x output 2 set register (HRTIM_SETx2R) (x = A to F)*

Table 409. HRTIM register memory map (continued)

Offset	Register name
Block A: 0x0C8 Block B: 0x148 Block C: 0x1C8 Block D: 0x248 Block E: 0x2C8 Block F: 0x348	*HRTIM timer x output 2 reset register (HRTIM_RSTx2R) (x = A to F)*
Block A: 0x0CC Block B: 0x14C Block C: 0x1CC Block D: 0x24C Block E: 0x2CC Block F: 0x34C	*HRTIM timer x external event filtering register 1 (HRTIM_EEFxR1) (x = A to F)*
Block A: 0x0D0 Block B: 0x150 Block C: 0x1D0 Block D: 0x250 Block E: 0x2D0 Block F: 0x350	*HRTIM timer x external event filtering register 2 (HRTIM_EEFxR2) (x = A to F)*
0x0D4	*HRTIM timer A reset register (HRTIM_RSTAR)*
0x154	*HRTIM timer B reset register (HRTIM_RSTBR)*
0x1D4	*HRTIM timer C reset register (HRTIM_RSTCR)*
0x254	*HRTIM timer D reset register (HRTIM_RSTDR)*
0x2D4	*HRTIM timer E reset register (HRTIM_RSTER)*
0x354	*HRTIM timer F reset register (HRTIM_RSTFR)*
Block A: 0x0D8 Block B: 0x158 Block C: 0x1D8 Block D: 0x258 Block E: 0x2D8 Block F: 0x358	*HRTIM timer x chopper register (HRTIM_CHPxR) (x = A to F)*
0x0DC	*HRTIM timer A capture 1 control register (HRTIM_CPT1ACR)*
0x15C	*HRTIM timer B capture 1 control register (HRTIM_CPT1BCR)*
0x1DC	*HRTIM timer C capture 1 control register (HRTIM_CPT1CCR)*
0x25C	*HRTIM timer D capture 1 control register (HRTIM_CPT1DCR)*
0x2DC	*HRTIM timer E capture 1 control register (HRTIM_CPT1ECR)*
0x35C	*HRTIM timer F capture 1 control register (HRTIM_CPT1FCR)*

Table 409. HRTIM register memory map (continued)

Offset	Register name
Block A: 0x0E0 Block B: 0x160 Block C: 0x1E0 Block D: 0x260 Block E: 0x2E0 Block F: 0x360	*HRTIM timer x capture 2 control register (HRTIM_CPT2xCR) (x = A to F)*
Block A: 0x0E4 Block B: 0x164 Block C: 0x1E4 Block D: 0x264 Block E: 0x2E4 Block F: 0x364	*HRTIM timer x output register (HRTIM_OUTxR) (x = A to F)*
Block A: 0x0E8 Block B: 0x168 Block C: 0x1E8 Block D: 0x268 Block E: 0x2E8 Block F: 0x368	*HRTIM timer x fault register (HRTIM_FLTxR) (x = A to F)*
Block A: 0x0EC Block B: 0x16C Block C: 0x1EC Block D: 0x26C Block E: 0x2EC Block F: 0x36C	*HRTIM timer x control register 2 (HRTIM_TIMxCR2) (x = A to F)*
Block A: 0x0F0 Block B: 0x170 Block C: 0x1F0 Block D: 0x270 Block E: 0x2F0 Block F: 0x370	*HRTIM timer x external event filtering register 3 (HRTIM_EEFxR3) (x = A to F)*
0x380	*HRTIM control register 1 (HRTIM_CR1)*
0x384	*HRTIM control register 2 (HRTIM_CR2)*
0x388	*HRTIM interrupt status register (HRTIM_ISR)*
0x38C	*HRTIM interrupt clear register (HRTIM_ICR)*
0x390	*HRTIM interrupt enable register (HRTIM_IER)*
0x394	*HRTIM output enable register (HRTIM_OENR)*
0x398	*HRTIM output disable register (HRTIM_ODISR)*
0x39C	*HRTIM output disable status register (HRTIM_ODSR)*
0x3A0	*HRTIM burst mode control register (HRTIM_BMCR)*
0x3A4	*HRTIM burst mode trigger register (HRTIM_BMTRGR)*
0x3A8	*HRTIM burst mode compare register (HRTIM_BMCMPR)*

Table 409. HRTIM register memory map (continued)

Offset	Register name
0x3AC	*HRTIM burst mode period register (HRTIM_BMPER)*
0x3B0	*HRTIM timer external event control register 1 (HRTIM_EECR1)*
0x3B4	*HRTIM timer external event control register 2 (HRTIM_EECR2)*
0x3B8	*HRTIM timer external event control register 3 (HRTIM_EECR3)*
0x3BC	*HRTIM ADC trigger 1 register (HRTIM_ADC1R)*
0x3C0	*HRTIM ADC trigger 2 register (HRTIM_ADC2R)*
0x3C4	*HRTIM ADC trigger 3 register (HRTIM_ADC3R)*
0x3C8	*HRTIM ADC trigger 4 register (HRTIM_ADC4R)*
0x3CC	*HRTIM DLL control register (HRTIM_DLLCR)*
0x3D0	*HRTIM fault input register 1 (HRTIM_FLTINR1)*
0x3D4	*HRTIM fault input register 2 (HRTIM_FLTINR2)*
0x3D8	*HRTIM burst DMA master timer update register (HRTIM_BDMUPR)*
Block A: 0x3DC Block B: 0x3E0 Block C: 0x3E4 Block D: 0x3E8 Block E: 0x3EC Block F: 0x3F4	*HRTIM burst DMA timer x update register (HRTIM_BDTxUPR) (x = A to F)*
0x3F0	*HRTIM burst DMA data register (HRTIM_BDMADR)*
0x3F8	*HRTIM ADC extended trigger register (HRTIM_ADCER)*
0x3FC	*HRTIM ADC trigger update register (HRTIM_ADCUR)*
0x400	*HRTIM ADC post scaler register 1 (HRTIM_ADCPS1)*
0x404	*HRTIM ADC post scaler register 2 (HRTIM_ADCPS2)*
0x408	*HRTIM fault input register 3 (HRTIM_FLTINR3)*
0x40C	*HRTIM fault input register 4 (HRTIM_FLTINR4)*

34.5.2 HRTIM master timer control register (HRTIM_MCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	38	27	26	25	24	23	22	21	20	19	18	17	16
BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	TFCEN	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN
rw	rw	rw		rw	rw	rw			rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		INTLVD[1:0]		HALF	RETRIG	CONT	CKPSC[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **BRSTDMA[1:0]**: Burst DMA update

These bits define how the update occurs relatively to a burst DMA transaction.

00: Update done independently from the DMA burst transfer completion

01: Update done when the DMA burst transfer is completed

10: Update done on master timer roll-over following a DMA burst transfer completion. This mode only works in continuous mode.

11: Reserved

Bit 29 **MREPU**: Master timer repetition update

This bit defines whether an update occurs when the master timer repetition period is completed (either due to roll-over or reset events). MREPU can be set only if BRSTDMA[1:0] = 00 or 01.

0: Update on repetition disabled

1: Update on repetition enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **PREEN**: Preload enable

This bit enables the registers preload mechanism and defines whether the write accesses to the memory mapped registers are done into HRTIM's active or preload registers.

0: Preload disabled: the write access is directly done into the active register

1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]** DAC synchronization

A DAC synchronization event can be enabled and generated when the master timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 34.3.21: DAC triggers* for connections details).

00: No DAC trigger generated

01: Trigger generated on hrtim_dac_trg1

10: Trigger generated on hrtim_dac_trg2

11: Trigger generated on hrtim_dac_trg3

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **TFCEN**: Timer F counter enable

This bit starts the timer F counter.

0: Timer F counter disabled

1: Timer F counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 21 **TECEN**: Timer E counter enable

This bit starts the timer E counter.

0: Timer E counter disabled

1: Timer E counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 20 **TDCEN**: Timer D counter enable

This bit starts the timer D counter.

0: Timer D counter disabled

1: Timer D counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 19 **TCCEN**: Timer C counter enable

This bit starts the timer C counter.

0: Timer C counter disabled

1: Timer C counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 18 **TBCEN**: Timer B counter enable
This bit starts the timer B counter.
0: Timer B counter disabled
1: Timer B counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 17 **TACEN**: Timer A counter enable
This bit starts the timer A counter.
0: Timer A counter disabled
1: Timer A counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 16 **MCEN**: Master timer counter enable
This bit starts the master timer counter.
0: Master counter disabled
1: Master counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bits 15:14 **SYNCSRC[1:0]**: Synchronization source
These bits define the source and event to be sent on the synchronization outputs SYNCOUT[2:1]
00: Master timer start
01: Master timer compare 1 event
10: Timer A start/reset
11: Timer A compare 1 event

Bits 13:12 **SYNCOUT[1:0]**: Synchronization output
These bits define the routing and conditioning of the synchronization output event.
00: Disabled
01: On-chip HRTIM(s). Allows the simultaneous start of several HRTIM instances.
10: Positive pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
11: Negative pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
Note: This bit field must not be modified once the counter is enabled (TxCEN bit set)

Bit 11 **SYNCSTRTM**: Synchronization starts master
This bit enables the master timer start when receiving a synchronization input event:
0: No effect on the master timer
1: A synchronization input event starts the master timer

Bit 10 **SYNCRSTM**: Synchronization resets master
This bit enables the master timer reset when receiving a synchronization input event:
0: No effect on the master timer
1: A synchronization input event resets the master timer

Bits 9:8 **SYNCIN[1:0]** Synchronization input
These bits are defining the synchronization input source.
00: Disabled. HRTIM is not synchronized and runs in standalone mode.
01: Internal HRTIM event (available only when multiple HRTIMs are instantiated).
HRTimer1: reserved
HRTimer2: hrtim1_scout1
10: Internal event on hrtim_in_sync[2]: the HRTIM is synchronized with the on-chip timer (refer to *Synchronization input*).
11: External event on hrtim_in_sync[3]: a positive pulse on HRTIM_SCIN input triggers the HRTIM.
Note: This parameter cannot be changed once the impacted timers are enabled.

Bits 7:6 **INTLVD[1:0]**: Interleaved mode

This bit field is significant only when the HALF bit is reset. It enables the interleaved mode.

00: Interleaved mode disabled

01: Triple interleaved mode: when HRTIM_MPER register is written, the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/3 value, and the HRTIM_MCMP2R active register is automatically updated with 2x (HRTIM_MPER/3) value.

10: Quad interleaved mode: when HRTIM_MPER register is written, the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/4 value, the HRTIM_MCMP2R active register is automatically updated with HRTIM_MPER/2 value and the HRTIM_MCMP3R active register is automatically updated with 3x (HRTIM_MPER/4) value.

11: Interleaved mode disabled

Bit 5 **HALF**: Half mode

This bit enables the half duty-cycle mode: the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/2 value when HRTIM_MPER register is written.

0: Half mode disabled

1: Half mode enabled

Bit 4 **RETRIG**: Re-triggerable mode

This bit defines the behavior of the master timer counter in single-shot mode.

0: The timer is not re-triggerable: a counter reset can be done only if the counter is stopped (period elapsed)

1: The timer is re-triggerable: a counter reset is done whatever the counter state (running or stopped)

Bit 3 **CONT**: Continuous mode

0: The timer operates in single-shot mode and stops when it reaches the MPER value

1: The timer operates in continuous (free-running) mode and rolls over to zero when it reaches the MPER value

Bits 2:0 **CKPSC[2:0]**: Clock prescaler

These bits define the master timer high-resolution clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{CKPSC[2:0]}$.

The prescaling ratio cannot be modified once the timer is enabled.

34.5.3 HRTIM master timer interrupt status register (HRTIM_MISR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPD	SYNC	MREP	MCMP 4	MCMP 3	MCMP 2	MCMP 1
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPD**: Master update interrupt flag

This bit is set by hardware when the master timer registers are updated.

0: No master update interrupt occurred

1: Master update interrupt occurred

Bit 5 **SYNC**: Sync input interrupt flag
This bit is set by hardware when a synchronization input event is received.
0: No sync input interrupt occurred
1: Sync input interrupt occurred

Bit 4 **MREP**: Master repetition interrupt flag
This bit is set by hardware when the master timer repetition period has elapsed.
0: No master repetition interrupt occurred
1: Master repetition interrupt occurred

Bit 3 **MCMP4**: Master compare 4 interrupt flag
Refer to MCMP1 description

Bit 2 **MCMP3**: Master compare 3 interrupt flag
Refer to MCMP1 description

Bit 1 **MCMP2**: Master compare 2 interrupt flag
Refer to MCMP1 description

Bit 0 **MCMP1**: Master compare 1 interrupt flag
This bit is set by hardware when the master timer counter matches the value programmed in the master compare 1 register.
0: No master compare 1 interrupt occurred
1: Master compare 1 interrupt occurred

34.5.4 HRTIM master timer interrupt clear register (HRTIM_MICR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPDC**: Master update interrupt flag clear
Writing 1 to this bit clears the MUPDC flag in HRTIM_MISR register.

Bit 5 **SYNCC**: Sync input interrupt flag clear
Writing 1 to this bit clears the SYNC flag in HRTIM_MISR register.

Bit 4 **MREPC**: Repetition interrupt flag clear
Writing 1 to this bit clears the MREP flag in HRTIM_MISR register.

Bit 3 **MCMP4C**: Master compare 4 interrupt flag clear
Writing 1 to this bit clears the MCMP4 flag in HRTIM_MISR register.

Bit 2 **MCMP3C**: Master compare 3 interrupt flag clear
Writing 1 to this bit clears the MCMP3 flag in HRTIM_MISR register.

Bit 1 **MCMP2C**: Master compare 2 interrupt flag clear
Writing 1 to this bit clears the MCMP2 flag in HRTIM_MISR register.

Bit 0 **MCMP1C**: Master compare 1 interrupt flag clear
Writing 1 to this bit clears the MCMP1 flag in HRTIM_MISR register.

34.5.5 HRTIM master timer DMA interrupt enable register (HRTIM_MDIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **MUPDDE**: Master update DMA request enable
This bit is set and cleared by software to enable/disable the master update DMA requests.
0: Master update DMA request disabled
1: Master update DMA request enabled

Bit 21 **SYNCDE**: Sync input DMA request enable
This bit is set and cleared by software to enable/disable the sync input DMA requests.
0: Sync input DMA request disabled
1: Sync input DMA request enabled

Bit 20 **MREPDE**: Master repetition DMA request enable
This bit is set and cleared by software to enable/disable the master timer repetition DMA requests.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **MCMP4DE**: Master compare 4 DMA request enable
Refer to MCMP1DE description

Bit 18 **MCMP3DE**: Master compare 3 DMA request enable
Refer to MCMP1DE description

Bit 17 **MCMP2DE**: Master compare 2 DMA request enable
Refer to MCMP1DE description

Bit 16 **MCMP1DE**: Master compare 1 DMA request enable
This bit is set and cleared by software to enable/disable the master timer compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

Bits 15:6 Reserved, must be kept at reset value.

Bit 6 **MUPDIE**: Master update interrupt enable
This bit is set and cleared by software to enable/disable the master timer registers update interrupts
0: Master update interrupts disabled
1: Master update interrupts enabled

Bit 5 **SYNCIE**: Sync input interrupt enable
This bit is set and cleared by software to enable/disable the sync input interrupts
0: Sync input interrupts disabled
1: Sync input interrupts enabled

Bit 4 **MREPIE**: Master repetition interrupt enable
This bit is set and cleared by software to enable/disable the master timer repetition interrupts
0: Master repetition interrupt disabled
1: Master repetition interrupt enabled

Bit 3 **MCMP4IE**: Master compare 4 interrupt enable
Refer to MCMP1IE description

Bit 2 **MCMP3IE**: Master compare 3 interrupt enable
Refer to MCMP1IE description

Bit 1 **MCMP2IE**: Master compare 2 interrupt enable
Refer to MCMP1IE description

Bit 0 **MCMP1IE**: Master compare 1 interrupt enable
This bit is set and cleared by software to enable/disable the master timer compare 1 interrupt
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

34.5.6 HRTIM master timer counter register (HRTIM_MCNTR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCNT[15:0]**: Counter value
Holds the master timer counter value. This register can only be written when the master timer is stopped (MCEN = 0 in HRTIM_MCR).

Note: For HR clock prescaling ratio below 32 (CKPSCCKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

Note: The timer behavior is not guaranteed if the counter value is set above the HRTIM_MPER register value.

34.5.7 HRTIM master timer period register (HRTIM_MPER)

Address offset: 0x014

Reset value: 0x0000 FFDF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPER[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MPER[15:0]**: Master timer period value

This register defines the counter overflow value.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The maximum value is 0x0000 FFDF.

34.5.8 HRTIM master timer repetition register (HRTIM_MREP)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MREP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **MREP[7:0]**: Master timer repetition period value

This register holds the repetition period value for the master counter. It is either the preload register or the active register if preload is disabled.

34.5.9 HRTIM master timer compare 1 register (HRTIM_MCMP1R)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP1[15:0]**: Master timer compare 1 value

This register holds the master timer compare 1 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

34.5.10 HRTIM master timer compare 2 register (HRTIM_MCMP2R)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP2[15:0]**: Master timer compare 2 value

This register holds the master timer compare 2 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

34.5.11 HRTIM master timer compare 3 register (HRTIM_MCMP3R)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP3[15:0]**: Master timer compare 3 value

This register holds the master timer compare 3 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

34.5.12 HRTIM master timer compare 4 register (HRTIM_MCMP4R)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP4[15:0]**: Master timer compare 4 value

This register holds the master timer compare 4 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

34.5.13 HRTIM timer x control register (HRTIM_TIMxCR) (x = A to F)

Address offset: Block A: 0x080

Address offset: Block B: 0x100

Address offset: Block C: 0x180

Address offset: Block D: 0x200

Address offset: Block E: 0x280

Address offset: Block F: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDGAT[3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	TAU	TxRSTU	TxREPU	TFU
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	RSYNCU	INTLVD[1:0]		PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **UPDGAT[3:0]**: Update gating

These bits define how the update occurs relatively to the burst DMA transaction and the external update request on update enable inputs hrtim_upd_en[3:1] (refer to *Table 380: Update enable inputs and sources*).

The update events, as mentioned below, can be: MSTU, TFU, TEU, TDU, TCU, TBU, TAU, TxRSTU, TxREPU.

0000: The update occurs independently from the DMA burst transfer
0001: The update occurs when the DMA burst transfer is completed
0010: The update occurs on the update event following the DMA burst transfer completion
0011: The update occurs on a rising edge on hrtim_upd_en1
0100: The update occurs on a rising edge on hrtim_upd_en2
0101: The update occurs on a rising edge on hrtim_upd_en3
0110: The update occurs on the update event following a rising edge on hrtim_upd_en1
0111: The update occurs on the update event following a rising edge on hrtim_upd_en2
1000: The update occurs on the update event following a rising edge on hrtim_upd_en3
Others: Reserved

Note: This bit field must be reset before programming a new value.
For UPDGAT[3:0] values equal to 0001, 0011, 0100, 0101, it is possible to have multiple concurrent update source (for instance RSTU and DMA burst).

Bit 27 **PREEN**: Preload enable

This bit enables the registers preload mechanism and defines whether a write access into a preloadable register is done into the active or the preload register.

0: Preload disabled: the write access is directly done into the active register
1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]** DAC synchronization

A DAC synchronization event is generated when the timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 34.3.21: DAC triggers* for connections details).

00: No DAC trigger generated
01: Trigger generated on hrtim_dac_trg1
10: Trigger generated on hrtim_dac_trg2
11: Trigger generated on hrtim_dac_trg3

Bit 24 **MSTU**: Master timer update

Register update is triggered by the master timer update.

0: Update by master timer disabled
1: Update by master timer enabled

Bit 23 **TEU**: Timer E update

Register update is triggered by the timer E update

0: Update by timer E disabled
1: Update by timer E enabled

Note: This bit is reserved for HRTIM_TIMECR. *It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMFCR.*

Bit 22 **TDU**: Timer D update

Register update is triggered by the timer D update

0: Update by timer D disabled
1: Update by timer D enabled

Note: This bit is reserved for HRTIM_TIMDCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 21 **TCU**: Timer C update

Register update is triggered by the timer C update

0: Update by timer C disabled

1: Update by timer C enabled

Note: This bit is reserved for HRTIM_TIMCCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 20 **TBU**: Timer B update

Register update is triggered by the timer B update

0: Update by timer B disabled

1: Update by timer B enabled

Note: This bit is reserved for HRTIM_TIMBCR. It is only available for HRTIM_TIMACR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 19 **TAU**: Timer A update

Register update is triggered by the timer A update

0: Update by timer A disabled

1: Update by timer A enabled

Note: This bit is reserved for HRTIM_TIMBCR. It is only available for HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 18 **TxRSTU**: Timer x reset update

Register update is triggered by timer x counter reset or roll-over to 0 after reaching the period value in continuous mode.

0: Update by timer x reset / roll-over disabled

1: Update by timer x reset / roll-over enabled

Bit 17 **TxREPU**: Timer x repetition update

Register update is triggered when the counter rolls over and HRTIM_REPx = 0

0: Update on repetition disabled

1: Update on repetition enabled

Bit 16 **TFU**: Timer F update

Register update is triggered by the timer F update

0: Update by timer F disabled

1: Update by timer F enabled

Note: This bit is reserved for HRTIM_TIMFCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR.

Bits 15:14 **DELCMP4[1:0]**: CMP4 auto-delayed mode

This bit field defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (refer to *Auto-delayed mode*).

00: CMP4 register is always active (standard compare mode)

01: CMP4 value is recomputed and is active following a capture 2 event

10: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 1 match (timeout function if capture 2 event is missing)

11: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 3 match (timeout function if capture event is missing)

Note: This bit field must not be modified once the counter is enabled (TxCEN bit set).

Bits 13:12 **DELCMP2[1:0]**: CMP2 auto-delayed mode

This bit field defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (refer to *Auto-delayed mode*).

00: CMP2 register is always active (standard compare mode)

01: CMP2 value is recomputed and is active following a capture 1 event

10: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after compare 1 match (timeout function if capture event is missing)

11: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after compare 3 match (timeout function if capture event is missing)

Note: This bit field must not be modified once the counter is enabled (TxCEN bit set).

Bit 11 **SYNCSTRTx**: Synchronization starts timer x

This bit defines the timer x behavior following the synchronization event:

0: No effect on timer x

1: A synchronization input event starts the timer x

Bit 10 **SYNCRSTx**: Synchronization resets timer x

This bit defines the timer x behavior following the synchronization event:

0: No effect on timer x

1: A synchronization input event resets the timer x

Bit 9 **RSYNCU**: Re-synchronized update

This bit specifies whether update source coming outside from the timing unit must be synchronized:

0: The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) is taken into account immediately

1: The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) is taken into account on the following reset/roll-over event.

Note: This bit is significant only when UPDGAT[3:0] = 0000, it is ignored otherwise.

Bits 8:7 **INTLVD[1:0]**: Interleaved mode

This bit field is significant only when the HALF bit is reset. It enables the interleaved mode.

00: Interleaved mode disabled

01: Triple interleaved mode: when HRTIM_PERxR register is written, the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/3 value, and the HRTIM_CMP2xR active register is automatically updated with 2x (HRTIM_PERxR/3) value.

10: Quad interleaved mode: when HRTIM_PERxR register is written, the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/4 value, the HRTIM_CMP2xR active register is automatically updated with HRTIM_PERxR/2 value and the HRTIM_CMP3xR active register is automatically updated with 3x (HRTIM_PERxR/4) value.

11: Interleaved mode disabled

Bit 6 **PSHPLL**: Push-pull mode enable

This bit enables the push-pull mode.

0: Push-pull mode disabled

1: Push-pull mode enabled

Note: This bit field must not be modified once the counter is enabled (TxCEN bit set).

Bit 5 **HALF**: Half mode enable

This bit enables the half duty-cycle mode: the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/2 value when HRTIM_PERxR register is written.

0: Half mode disabled

1: Half mode enabled

Bit 4 **RETRIG**: Re-triggerable mode

This bit defines the counter behavior in single shot mode.

0: The timer is not re-triggerable: a counter reset is done if the counter is stopped (period elapsed in single-shot mode or counter stopped in continuous mode)

1: The timer is re-triggerable: a counter reset is done whatever the counter state.

Bit 3 **CONT**: Continuous mode

This bit defines the timer operating mode.

0: The timer operates in single-shot mode and stops when it reaches TIMxPER value

1: The timer operates in continuous mode and rolls over to zero when it reaches TIMxPER value

Bits 2:0 **CKPSCx[2:0]**: HRTIM timer x clock prescaler

These bits define the master timer high-resolution clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{CKPSC[2:0]}$.

The prescaling ratio cannot be modified once the timer is enabled.

34.5.14 HRTIM timer x interrupt status register (HRTIM_TIMxISR) (x = A to F)

Address offset: Block A: 0x084

Address offset: Block B: 0x104

Address offset: Block C: 0x184

Address offset: Block D: 0x204

Address offset: Block E: 0x284

Address offset: Block F: 0x304

Reset value: 0x000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	O2CPY	O1CPY	O2 STAT	O1 STAT	IPP STAT	CPP STAT
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	Res.	REP	CMP4	CMP3	CMP2	CMP1
	r	r	r	r	r	r	r	r	r		r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **O2CPY**: Output 2 copy

This status bit is a raw copy of the output 2 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 2 is inactive

1: Output 2 is active

Bit 20 **O1CPY**: Output 1 copy

This status bit is a raw copy of the output 1 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 1 is inactive

1: Output 1 is active

Bit 19 **O2STAT**: Output 2 status
This status bit indicates the output 2 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.
0: Output 2 was inactive
1: Output 2 was active

Bit 18 **O1STAT**: Output 1 status
This status bit indicates the output 1 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.
0: Output 1 was inactive
1: Output 1 was active

Bit 17 **IPPSTAT**: Idle push-pull Status
This status bit indicates on which output the signal was applied, in push-pull mode balanced fault mode or delayed idle mode, when the protection was triggered (whatever the output state, active or inactive).
0: Protection occurred when the output 1 was active and output 2 forced inactive
1: Protection occurred when the output 2 was active and output 1 forced inactive

Bit 16 **CPPSTAT**: Current push-pull status
This status bit indicates on which output the signal is currently applied, in push-pull mode. It is only significant in this configuration.
0: Signal applied on output 1 and output 2 forced inactive
1: Signal applied on output 2 and output 1 forced inactive

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DLYPRT**: Delayed protection flag
0: No delayed protection interrupt occured
1: Delayed Idle or balanced Idle mode entry occured

Bit 13 **RST**: Reset and/or roll-over interrupt flag
This bit is set by hardware when the timer x counter is reset or rolls over in continuous mode.
0: No TIMx counter reset/roll-over interrupt occurred
1: TIMX counter reset/roll-over interrupt occurred

Bit 12 **RSTx2**: Output 2 reset interrupt flag
Refer to RSTx1 description

Bit 11 **SETx2**: Output 2 set interrupt flag
Refer to SETx1 description

Bit 10 **RSTx1**: Output 1 reset interrupt flag
This bit is set by hardware when the Tx1 output is reset (goes from active to inactive mode).
0: No Tx1 output reset interrupt occurred
1: Tx1 output reset interrupt occurred

Bit 9 **SETx1**: Output 1 set interrupt flag
This bit is set by hardware when the Tx1 output is set (goes from inactive to active mode).
0: No Tx1 output set interrupt occurred
1: Tx1 output set interrupt occurred

Bit 8 **CPT2**: Capture 2 interrupt flag
Refer to CPT1 description

Bit 7 **CPT1**: Capture 1 interrupt flag
This bit is set by hardware when the timer x capture 1 event occurs.
0: No timer x capture 1 interrupt occurred
1: Timer x capture 1 interrupt occurred

Bit 6 **UPD**: Update interrupt flag
This bit is set by hardware when the timer x update event occurs.
0: No timer x update interrupt occurred
1: Timer x update interrupt occurred

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REP**: Repetition interrupt flag
This bit is set by hardware when the timer x repetition period has elapsed.
0: No timer x repetition interrupt occurred
1: Timer x repetition interrupt occurred

Bit 3 **CMP4**: Compare 4 interrupt flag
Refer to CMP1 description

Bit 2 **CMP3**: Compare 3 interrupt flag
Refer to CMP1 description

Bit 1 **CMP2**: Compare 2 interrupt flag
Refer to CMP1 description

Bit 0 **CMP1**: Compare 1 interrupt flag
This bit is set by hardware when the timer x counter matches the value programmed in the compare 1 register.
0: No compare 1 interrupt occurred
1: Compare 1 interrupt occurred

34.5.15 HRTIM timer x interrupt clear register (HRTIM_TIMxICR) (x = A to F)

Address offset: Block A: 0x088

Address offset: Block B: 0x108

Address offset: Block C: 0x188

Address offset: Block D: 0x208

Address offset: Block E: 0x288

Address offset: Block F: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYPRTC	RSTC	RSTx2C	SET2xC	RSTx1C	SET1xC	CPT2C	CPT1C	UPDC	Res.	REPC	CMP4C	CMP3C	CMP2C	CMP1C
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **DLYPRTC**: Delayed protection flag clear
Writing 1 to this bit clears the DLYPRT flag in HRTIM_TIMxISR register

Bit 13 **RSTC**: Reset interrupt flag clear
Writing 1 to this bit clears the RST flag in HRTIM_TIMxISR register

Bit 12 **RSTx2C**: Output 2 reset flag clear
Writing 1 to this bit clears the RSTx2 flag in HRTIM_TIMxISR register

Bit 11 **SETx2C**: Output 2 set flag clear
Writing 1 to this bit clears the SETx2 flag in HRTIM_TIMxISR register

Bit 10 **RSTx1C**: Output 1 reset flag clear
Writing 1 to this bit clears the RSTx1 flag in HRTIM_TIMxISR register

Bit 9 **SETx1C**: Output 1 set flag clear
Writing 1 to this bit clears the SETx1 flag in HRTIM_TIMxISR register

Bit 8 **CPT2C**: Capture 2 interrupt flag clear
Writing 1 to this bit clears the CPT2 flag in HRTIM_TIMxISR register

Bit 7 **CPT1C**: Capture 1 interrupt flag clear
Writing 1 to this bit clears the CPT1 flag in HRTIM_TIMxISR register

Bit 6 **UPDC**: Update interrupt flag clear
Writing 1 to this bit clears the UPD flag in HRTIM_TIMxISR register

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REPC**: Repetition interrupt flag clear
Writing 1 to this bit clears the REP flag in HRTIM_TIMxISR register

Bit 3 **CMP4C**: Compare 4 interrupt flag clear
Writing 1 to this bit clears the CMP4 flag in HRTIM_TIMxISR register

Bit 2 **CMP3C**: Compare 3 interrupt flag clear
Writing 1 to this bit clears the CMP3 flag in HRTIM_TIMxISR register

Bit 1 **CMP2C**: Compare 2 interrupt flag clear
Writing 1 to this bit clears the CMP2 flag in HRTIM_TIMxISR register

Bit 0 **CMP1C**: Compare 1 interrupt flag clear
Writing 1 to this bit clears the CMP1 flag in HRTIM_TIMxISR register

34.5.16 HRTIM timer x DMA interrupt enable register (HRTIM_TIMxDIER) (x = A to F)

Address offset: Block A: 0x08C

Address offset: Block B: 0x10C

Address offset: Block C: 0x18C

Address offset: Block D: 0x20C

Address offset: Block E: 0x28C

Address offset: Block F: 0x30C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SETx1DE	CPT2DE	CPT1DE	UPDDE	Res.	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SETx1IE	CPT2IE	CPT1IE	UPDIE	Res.	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DLYPRTDE**: Delayed protection DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on delayed protection.
0: Delayed protection DMA request disabled
1: Delayed protection DMA request enabled

Bit 29 **RSTDE**: Reset/roll-over DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over DMA request disabled
1: Timer x counter reset/roll-over DMA request enabled

Bit 28 **RSTx2DE**: Output 2 reset DMA request enable
Refer to RSTx1DE description

Bit 27 **SETx2DE**: Output 2 set DMA request enable
Refer to SETx1DE description

Bit 26 **RSTx1DE**: Output 1 reset DMA request enable
This bit is set and cleared by software to enable/disable Tx1 output reset DMA requests.
0: Tx1 output reset DMA request disabled
1: Tx1 output reset DMA request enabled

Bit 25 **SETx1DE**: Output 1 set DMA request enable
This bit is set and cleared by software to enable/disable Tx1 output set DMA requests.
0: Tx1 output set DMA request disabled
1: Tx1 output set DMA request enabled

Bit 24 **CPT2DE**: Capture 2 DMA request enable
Refer to CPT1DE description

Bit 23 **CPT1DE**: Capture 1 DMA request enable
This bit is set and cleared by software to enable/disable capture 1 DMA requests.
0: Capture 1 DMA request disabled
1: Capture 1 DMA request enabled

Bit 22 **UPDDE**: Update DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on update event.
0: Update DMA request disabled
1: Update DMA request enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **REPDE**: Repetition DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on repetition event.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **CMP4DE**: Compare 4 DMA request enable
Refer to CMP1DE description

Bit 18 **CMP3DE**: Compare 3 DMA request enable
Refer to CMP1DE description

Bit 17 **CMP2DE**: Compare 2 DMA request enable
Refer to CMP1DE description

Bit 16 **CMP1DE**: Compare 1 DMA request enable
This bit is set and cleared by software to enable/disable the compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DLYPRTIE**: Delayed protection interrupt enable
This bit is set and cleared by software to enable/disable interrupts on delayed protection.
0: Delayed protection interrupts disabled
1: Delayed protection interrupts enabled

Bit 13 **RSTIE**: Reset/roll-over interrupt enable
This bit is set and cleared by software to enable/disable interrupts on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over interrupt disabled
1: Timer x counter reset/roll-over interrupt enabled

Bit 12 **RSTx2IE**: Output 2 reset interrupt enable
Refer to RSTx1IE description

Bit 11 **SETx2IE**: Output 2 set interrupt enable
Refer to SETx1IE description

Bit 10 **RSTx1IE**: Output 1 reset interrupt enable
This bit is set and cleared by software to enable/disable Tx1 output reset interrupts.
0: Tx1 output reset interrupts disabled
1: Tx1 output reset interrupts enabled

Bit 9 **SETx1IE**: Output 1 set interrupt enable
This bit is set and cleared by software to enable/disable Tx1 output set interrupts.
0: Tx1 output set interrupts disabled
1: Tx1 output set interrupts enabled

Bit 8 **CPT2IE**: Capture interrupt enable
Refer to CPT1IE description

Bit 7 **CPT1IE**: Capture interrupt enable
This bit is set and cleared by software to enable/disable capture 1 interrupts.
0: Capture 1 interrupts disabled
1: Capture 1 interrupts enabled

Bit 6 **UPDIE**: Update interrupt enable
This bit is set and cleared by software to enable/disable update event interrupts.
0: Update interrupts disabled
1: Update interrupts enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REPIE**: Repetition interrupt enable
This bit is set and cleared by software to enable/disable repetition event interrupts.
0: Repetition interrupts disabled
1: Repetition interrupts enabled

Bit 3 **CMP4IE**: Compare 4 interrupt enable
Refer to CMP1IE description

Bit 2 **CMP3IE**: Compare 3 interrupt enable
Refer to CMP1IE description

Bit 1 **CMP2IE**: Compare 2 interrupt enable
Refer to CMP1IE description

Bit 0 **CMP1IE**: Compare 1 interrupt enable
This bit is set and cleared by software to enable/disable the compare 1 interrupts.
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

34.5.17 HRTIM timer x counter register (HRTIM_CNTxR) (x = A to F)

Address offset: Block A: 0x090

Address offset: Block B: 0x110

Address offset: Block C: 0x190

Address offset: Block D: 0x210

Address offset: Block E: 0x290

Address offset: Block F: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNTx[15:0]**: Timer x counter value

This register holds the timer x counter value. It can only be written when the timer is stopped (TxCEN = 0 in HRTIM_TIMxCR).

Note: For HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

Note: The timer behavior is not guaranteed if the counter value is above the HRTIM_PERxR register value.

34.5.18 HRTIM timer x period register (HRTIM_PERxR) (x = A to F)

Address offset: Block A: 0x094

Address offset: Block B: 0x114

Address offset: Block C: 0x194

Address offset: Block D: 0x214

Address offset: Block E: 0x294

Address offset: Block F: 0x314

Reset value: 0x0000 FFDF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PERx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PERx[15:0]**: Timer x period value

This register holds timer x period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The maximum value is 0x0000 FFDF.

34.5.19 HRTIM timer x repetition register (HRTIM_REPxR) (x = A to F)

Address offset: Block A: 0x098

Address offset: Block B: 0x118

Address offset: Block C: 0x198

Address offset: Block D: 0x218

Address offset: Block E: 0x298

Address offset: Block F: 0x318

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits31:8 Reserved, must be kept at reset value.

Bits 7:0 **REPx[7:0]**: Timer x repetition period value

This register holds the repetition period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

34.5.20 HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F)

Address offset: Block A: 0x09C

Address offset: Block B: 0x11C

Address offset: Block C: 0x19C

Address offset: Block D: 0x21C

Address offset: Block E: 0x29C

Address offset: Block F: 0x31C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP1x[15:0]**: Timer x compare 1 value

This register holds the compare 1 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The null value is programmed following the use case described in *Null duty cycle exception case*.

34.5.21 HRTIM timer x compare 1 compound register (HRTIM_CMP1CxR) (x = A to F)

Address offset: Block A: 0x0A0

Address offset: Block B: 0x120

Address offset: Block C: 0x1A0

Address offset: Block D: 0x220

Address offset: Block E: 0x2A0

Address offset: Block F: 0x320

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **REPx[7:0]**: Timer x repetition value (aliased from HRTIM_REPx register)

This bit field is an alias from the REPx[7:0] bit field in the HRTIMx_REPxR register.

Bits 15:0 **CMP1x[15:0]**: Timer x compare 1 value

This bit field is an alias from the CMP1x[15:0] bit field in the HRTIMx_CMP1xR register.

34.5.22 HRTIM timer x compare 2 register (HRTIM_CMP2xR) (x = A to F)

Address offset: Block A: 0x0A4

Address offset: Block B: 0x124

Address offset: Block C: 0x1A4

Address offset: Block D: 0x224

Address offset: Block E: 0x2A4

Address offset: Block F: 0x324

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP2x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP2x[15:0]**: Timer x compare 2 value

This register holds the compare 2 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

This register behaves as an auto-delayed compare register, if enabled with DELCMP2[1:0] bits in HRTIM_TIMxCR.

34.5.23 HRTIM timer x compare 3 register (HRTIM_CMP3xR) (x = A to F)

Address offset: Block A: 0x0A8

Address offset: Block B: 0x128

Address offset: Block C: 0x1A8

Address offset: Block D: 0x228

Address offset: Block E: 0x2A8

Address offset: Block F: 0x328

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP3x[15:0]**: Timer x compare 3 value

This register holds the compare 3 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the fHRTIM clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The null value is programmed following the use case described in *Null duty cycle exception case*.

34.5.24 HRTIM timer x compare 4 register (HRTIM_CMP4xR) (x = A to F)

Address offset: Block A: 0x0AC

Address offset: Block B: 0x12C

Address offset: Block C: 0x1AC

Address offset: Block D: 0x22C

Address offset: Block E: 0x2AC

Address offset: Block F: 0x32C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP4x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP4x[15:0]**: Timer x compare 4 value

This register holds the compare 4 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

This register can behave as an auto-delayed compare register, if enabled with DELCMP4[1:0] bits in HRTIM_TIMxCR.

34.5.25 HRTIM timer x capture 1 register (HRTIM_CPT1xR) (x = A to F)

Address offset: Block A: 0x0B0

Address offset: Block B: 0x130

Address offset: Block C: 0x1B0

Address offset: Block D: 0x230

Address offset: Block E: 0x2B0

Address offset: Block F: 0x330

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT1x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DIR**: Timer x capture 1 direction status
This register holds the counting direction value when the capture 1 event occurred:
0: timer is up-counting
1: timer is down-counting
In up-counting mode (UDM bit reset), the DIR bit is always read as 0.

Bits 15:0 **CPT1x[15:0]**: Timer x capture 1 value
This register holds the counter value when the capture 1 event occurred.

Note: In up/down mode (UDM bit set to 1), the capture value is referred to:
Counting reset, when up-counting,
The PER event when down counting.

In UDM mode, the counter is always incrementing: the DIR bit allows to discriminate the up-down phases when reading the captured value.

Note: This is a regular resolution register: for HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

34.5.26 HRTIM timer x capture 2 register (HRTIM_CPT2xR) (x = A to F)

Address offset: Block A: 0x0B4

Address offset: Block B: 0x134

Address offset: Block C: 0x1B4

Address offset: Block D: 0x234

Address offset: Block E: 0x2B4

Address offset: Block F: 0x334

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT2x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DIR**: Timer x capture 1 direction status
This register holds the counting direction value when the capture 1 event occurred:
0: timer is up-counting
1: timer is down-counting
In up-counting mode (UDM bit reset), the DIR bit is always read as 0.

Bits 15:0 **CPT2x[15:0]**: Timer x capture 2 value
This register holds the counter value when the capture 2 event occurred.

Note: In up/down mode (UDM bit set to 1), the capture value is referred to:
Counting reset, when up-counting,
The PER event when down counting.

The DIR bit allows to discriminate the up-down phases when reading the captured value.

Note: This is a regular resolution register: for HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

34.5.27 HRTIM timer x deadtime register (HRTIM_DTxR) (x = A to F)

Address offset: Block A: 0x0B8

Address offset: Block B: 0x138

Address offset: Block C: 0x1B8

Address offset: Block D: 0x238

Address offset: Block E: 0x2B8

Address offset: Block F: 0x338

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTFLKx	DTFSLKx	Res.	Res.	Res.	Res.	SDTFx	DTFx[8:0]								
rwo	rwo					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTRLKx	DTRSLKx	Res.	DTPRSC[1:0]			SDTRx	DTRx[8:0]								
rwo	rwo		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DTFLKx**: Deadtime falling lock

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.

0: Deadtime falling value and sign is writable

1: Deadtime falling value and sign is read-only

Note: This bit is not preloaded

Bit 30 **DTFSLKx**: Deadtime falling sign lock

This write-once bit prevents the sign of falling deadtime to be modified, if enabled.

0: Deadtime falling sign is writable

1: Deadtime falling sign is read-only

Note: This bit is not preloaded.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **SDTFx**: Sign deadtime falling value

This register determines whether the deadtime is positive (signals not overlapping) or negative (signals overlapping).

0: Positive deadtime on falling edge

1: Negative deadtime on falling edge

Bits 24:16 **DTFx[8:0]**: Deadtime falling value

This register holds the value of the deadtime following a falling edge of reference PWM signal.

t_{DTF} = DTFx[8:0] x t_{DTG}

Bit 15 **DTRLKx**: Deadtime rising lock

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.

0: Deadtime rising value and sign is writable

1: Deadtime rising value and sign is read-only

Note: This bit is not preloaded.

Bit 14 **DTRSLKx**: Deadtime rising sign lock
This write-once bit prevents the sign of deadtime to be modified, if enabled.
0: Deadtime rising sign is writable
1: Deadtime rising sign is read-only
Note: This bit is not preloaded.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DTPRSC[2:0]**: Deadtime prescaler
This register holds the value of the deadtime clock prescaler.
$t_{DTG} = (2^{(DTPRSC[2:0])}) \times (t_{HRTIM} / 8)$
Refer to *Table 388*.
This bit field is read-only as soon as any of the lock bit is enabled (DTFLKs, DTFSLKx, DTRLKx, DTRSLKx).

Bit 9 **SDTRx**: Sign deadtime rising value
This register determines whether the deadtime is positive or negative (overlapping signals).
0: Positive deadtime on rising edge
1: Negative deadtime on rising edge

Bits 8:0 **DTRx[8:0]**: Deadtime rising value
This register holds the value of the deadtime following a rising edge of reference PWM signal.
$t_{DTR} = DTRx[8:0] \times t_{DTG}$

34.5.28 HRTIM timer x output 1 set register (HRTIM_SETx1R) (x = A to F)

Address offset: Block A: 0x0BC

Address offset: Block B: 0x13C

Address offset: Block C: 0x1BC

Address offset: Block D: 0x23C

Address offset: Block E: 0x2BC

Address offset: Block F: 0x33C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer preload to active)
Register update event forces the output to its active state.

Bit 30 **EXTEVNT10**: External event 10
Refer to EXTEVNT1 description.

Bit 29 **EXTEVNT9**: External event 9
Refer to EXTEVNT1 description.

Bit 28 **EXTEVNT8**: External event 8
Refer to EXTEVNT1 description.

Bit 27 **EXTEVNT7**: External event 7
Refer to EXTEVNT1 description.

Bit 26 **EXTEVNT6**: External event 6
Refer to EXTEVNT1 description.

Bit 25 **EXTEVNT5**: External event 5
Refer to EXTEVNT1 description.

Bit 24 **EXTEVNT4**: External event 4
Refer to EXTEVNT1 description.

Bit 23 **EXTEVNT3**: External event 3
Refer to EXTEVNT1 description.

Bit 22 **EXTEVNT2**: External event 2
Refer to EXTEVNT1 description.

Bit 21 **EXTEVNT1**: External event 1
External event 1 forces the output to its active state.

Bit 20 **TIMEVNT9**: Timer event 9
Refer to TIMEVNT1 description.

Bit 19 **TIMEVNT8**: Timer event 8
Refer to TIMEVNT1 description.

Bit 18 **TIMEVNT7**: Timer event 7
Refer to TIMEVNT1 description.

Bit 17 **TIMEVNT6**: Timer event 6
Refer to TIMEVNT1 description.

Bit 16 **TIMEVNT5**: Timer event 5
Refer to TIMEVNT1 description.

Bit 15 **TIMEVNT4**: Timer event 4
Refer to TIMEVNT1 description.

Bit 14 **TIMEVNT3**: Timer event 3
Refer to TIMEVNT1 description.

Bit 13 **TIMEVNT2**: Timer event 2
Refer to TIMEVNT1 description.

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its active state (refer to *Table 385* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its active state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its active state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its active state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its active state.

Bit 7 **MSTPER**: Master period

The master timer counter roll-over in continuous mode, or to the master timer reset in single-shot mode forces the output to its active state.

Bit 6 **CMP4**: Timer x compare 4

Timer x compare 4 event forces the output to its active state.

Bit 5 **CMP3**: Timer x compare 3

Timer x compare 3 event forces the output to its active state.

Bit 4 **CMP2**: Timer x compare 2

Timer x compare 2 event forces the output to its active state.

Bit 3 **CMP1**: Timer x compare 1

Timer x compare 1 event forces the output to its active state.

Bit 2 **PER**: Timer x period

Timer x period event forces the output to its active state.

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer x resynchronization

Timer x reset event coming solely from software or SYNC input forces the output to its active state.

Note: Other timer resets are not affecting the output when RESYNC=1.

Bit 0 **SST**: Software set trigger

This bit forces the output to its active state. This bit can only be set by software and is reset by hardware.

Note: This bit is not preloaded.

34.5.29 HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F)

Address offset: Block A: 0x0C0

Address offset: Block B: 0x140

Address offset: Block C: 0x1C0

Address offset: Block D: 0x240

Address offset: Block E: 0x2C0

Address offset: Block F: 0x340

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT(1)
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

1. SRT = software reset trigger

Bits 31:0 Refer to HRTIM_SETx1R bits description.
These bits are defining the source which can force the Tx1 output to its inactive state.

34.5.30 HRTIM timer x output 2 set register (HRTIM_SETx2R) (x = A to F)

Address offset: Block A: 0x0C4

Address offset: Block B: 0x144

Address offset: Block C: 0x1C4

Address offset: Block D: 0x244

Address offset: Block E: 0x2C4

Address offset: Block F: 0x344

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.
These bits are defining the source which forces the Tx2 output to its active state.

34.5.31 HRTIM timer x output 2 reset register (HRTIM_RSTx2R) (x = A to F)

Address offset: Block A: 0x0C8

Address offset: Block B: 0x148

Address offset: Block C: 0x1C8

Address offset: Block D: 0x248

Address offset: Block E: 0x2C8

Address offset: Block F: 0x348

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT(1)
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

1. SRT = software reset trigger

Bits 31:0 Refer to HRTIM_SETx1R bits description.
These bits are defining the source which forces the Tx2 output to its inactive state.

34.5.32 HRTIM timer x external event filtering register 1 (HRTIM_EEFxR1) (x = A to F)

Address offset: Block A: 0x0CC

Address offset: Block B: 0x14C

Address offset: Block C: 0x1CC

Address offset: Block D: 0x24C

Address offset: Block E: 0x2CC

Address offset: Block F: 0x34C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE5FLTR[3:0]				EE5LTCH	Res.	EE4FLTR[3:0]				EE4LTCH	Res.	EE3FLTR[3]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE3FLTR[2:0]			EE3LTCH	Res.	EE2FLTR[3:0]				EE2LTCH	Res.	EE1FLTR[3:0]				EE1LTCH
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **EE5FLTR[3:0]**: External event 5 filter
Refer to EE1FLTR[3:0] description.

Bit 24 **EE5LTCH**: External event 5 latch
Refer to EE1LTCH description

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 **EE4FLTR[3:0]**: External event 4 filter
Refer to EE1FLTR[3:0] description.

Bit 18 **EE4LTCH**: External event 4 latch
Refer to EE1LTCH description

Bit 17 Reserved, must be kept at reset value.

Bits 16:13 **EE3FLTR[3:0]**: External event 3 filter
Refer to EE1FLTR[3:0] description.

Bit 12 **EE3LTCH**: External event 3 latch
Refer to EE1LTCH description

Bit 11 Reserved, must be kept at reset value.

Bits 10:7 **EE2FLTR[3:0]**: External event 2 filter
Refer to EE1FLTR[3:0] description.

Bit 6 **EE2LTCH**: External event 2 latch
Refer to EE1LTCH description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:1 **EE1FLTR[3:0]**: External event 1 filter

0000: No filtering
0001: Blanking from counter reset/roll-over to compare 1
0010: Blanking from counter reset/roll-over to compare 2 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): blanking from compare 1 to compare 2, only during the up-counting phase.
0011: Blanking from counter reset/roll-over to compare 3
0100: Blanking from counter reset/roll-over to compare 4
0100: Blanking from counter reset/roll-over to compare 4 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): blanking from compare 3 to compare 4, only during the up-counting phase.
0101: Blanking from another timing unit: TIMFLTR1 source (refer to *Table 393: Filtering signals mapping per timer* for details)
0110: Blanking from another timing unit: TIMFLTR2 source (refer to *Table 393: Filtering signals mapping per timer* for details)
0111: Blanking from another timing unit: TIMFLTR3 source (refer to *Table 393: Filtering signals mapping per timer* for details)
1000: Blanking from another timing unit: TIMFLTR4 source (refer to *Table 393: Filtering signals mapping per timer* for details)
1001: Blanking from another timing unit: TIMFLTR5 source (refer to *Table 393: Filtering signals mapping per timer* for details)
1010: Blanking from another timing unit: TIMFLTR6 source (refer to *Table 393: Filtering signals mapping per timer* for details)
1011: Blanking from another timing unit: TIMFLTR7 source (refer to *Table 393: Filtering signals mapping per timer* for details)
1100: Blanking from another timing unit: TIMFLTR8 source (refer to *Table 393: Filtering signals mapping per timer* for details)
1101: Windowing from counter reset/roll-over to compare 2 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 to compare 3, only during the up-counting phase.
1110: Windowing from counter reset/roll-over to compare 3 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 to compare 3, only during the down-counting phase.
1111: Windowing from another timing unit: TIMWIN source (refer to *Table 394: Windowing signals mapping per timer (EEFLTR[3:0] = 1111)* for details) in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 during the up-counting phase to compare 3 during the down-counting phase.

Note: Whenever a compare register is used for filtering, the value must be strictly above 0.
This bit field must not be modified once the counter is enabled (TxCEN bit set)

Bit 0 **EE1LTCH**: External event 1 latch

0: Event 1 is ignored if it happens during a blank, or passed through during a window.
1: Event 1 is latched and delayed till the end of the blanking or windowing period.

Note: A timeout event is generated in window mode (EE1FLTR[3:0]=1101, 1110, 1111) if EE1LTCH = 0, except if the external event is programmed in fast mode (EExFAST = 1).
This bit field must not be modified once the counter is enabled (TxCEN bit set).

34.5.33 HRTIM timer x external event filtering register 2 (HRTIM_EEFxR2) (x = A to F)

Address offset: Block A: 0x0D0

Address offset: Block B: 0x150

Address offset: Block C: 0x1D0

Address offset: Block D: 0x250

Address offset: Block E: 0x2D0

Address offset: Block F: 0x350

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10FLTR[3:0]				EE10LTCH	Res.	EE9FLTR[3:0]				EE9LTCH	Res.	EE8FLTR[3]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8FLTR[2:0]			EE8LTCH	Res.	EE7FLTR[3:0]				EE7LTCH	Res.	EE6FLTR[3:0]				EE6LTCH
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **EE10FLTR[3:0]**: External event 10 filter
Refer to EE1FLTR[3:0] description.

Bit 24 **EE10LTCH**: External event 10 latch
Refer to EE1LTCH description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 **EE9FLTR[3:0]**: External event 9 filter
Refer to EE1FLTR[3:0] description.

Bit 18 **EE9LTCH**: External event 9 latch
Refer to EE1LTCH description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:13 **EE8FLTR[3:0]**: External event 8 filter
Refer to EE1FLTR[3:0] description.

Bit 12 **EE8LTCH**: External event 8 latch
Refer to EE1LTCH description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:7 **EE7FLTR[3:0]**: External event 7 filter
Refer to EE1FLTR[3:0] description.

Bit 6 **EE7LTCH**: External event 7 latch
Refer to EE1LTCH description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:1 **EE6FLTR[3:0]**: External event 6 filter
Refer to EE1FLTR[3:0] description.

Bit 0 **EE6LTCH**: External event 6 latch
Refer to EE1LTCH description.

34.5.34 HRTIM timer A reset register (HRTIM_RSTAR)

Address offset: 0x0D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIME CMP4	TIME CMP2	TIME CMP1	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2
The timer A counter is reset upon timer F Compare 2 event.

Bit 30 TIME**CPM4**: Timer E compare 4
The timer A counter is reset upon timer E compare 4 event.

Bit 29 **T**IM**ECMP2**: Timer E compare 2
The timer A counter is reset upon timer E compare 2 event.

Bit 28 **T**IM**ECMP1**: Timer E compare 1
The timer A counter is reset upon timer E compare 1 event.

Bit 27 **T**IM**DCMP4**: Timer D compare 4
The timer A counter is reset upon timer D compare 4 event.

Bit 26 **T**IM**DCMP2**: Timer D compare 2
The timer A counter is reset upon timer D compare 2 event.

Bit 25 **T**IM**DCMP1**: Timer D compare 1
The timer A counter is reset upon timer D compare 1 event.

Bit 24 **T**IM**CCMP4**: Timer C compare 4
The timer A counter is reset upon timer C compare 4 event.

Bit 23 **T**IM**CCMP2**: Timer C compare 2
The timer A counter is reset upon timer C compare 2 event.

Bit 22 **T**IM**CCMP1**: Timer C compare 1
The timer A counter is reset upon timer C compare 1 event.

Bit 21 **T**IM**BCMP4**: Timer B compare 4
The timer A counter is reset upon timer B compare 4 event.

Bit 20 **T**IM**BCMP2**: Timer B compare 2
The timer A counter is reset upon timer B compare 2 event.

Bit 19 **TIMBCMP1**: Timer B compare 1
The timer A counter is reset upon timer B compare 1 event.

Bit 18 **EXTEVNT10**: External event
The timer A counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9
The timer A counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer A counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer A counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer A counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer A counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer A counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer A counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer A counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer A counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer A counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer A counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer A counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer A counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer A counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer A compare 4 reset
The timer A counter is reset upon timer A compare 4 event.

Bit 2 **CMP2**: Timer A compare 2 reset
The timer A counter is reset upon timer A compare 2 event.

Bit 1 **UPDT**: Timer A update reset
The timer A counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer A counter is reset upon timer F compare 1 event.

34.5.35 HRTIM timer B reset register (HRTIM_RSTBR)

Address offset: 0x154

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIME CMP4	TIME CMP2	TIME CMP1	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2
The timer B counter is reset upon timer F Compare 2 event.

Bit 30 TIME**CPM4**: Timer E compare 4
The timer B counter is reset upon timer E compare 4 event.

Bit 29 **T**IM**ECMP2**: Timer E compare 2
The timer B counter is reset upon timer E compare 2 event.

Bit 28 **T**IM**ECMP1**: Timer E compare 1
The timer B counter is reset upon timer E compare 1 event.

Bit 27 **T**IM**DCMP4**: Timer D compare 4
The timer B counter is reset upon timer D compare 4 event.

Bit 26 **T**IM**DCMP2**: Timer D compare 2
The timer B counter is reset upon timer D compare 2 event.

Bit 25 **T**IM**DCMP1**: Timer D compare 1
The timer B counter is reset upon timer D compare 1 event.

Bit 24 **T**IM**CCMP4**: Timer C compare 4
The timer B counter is reset upon timer C compare 4 event.

Bit 23 **T**IM**CCMP2**: Timer C compare 2
The timer B counter is reset upon timer C compare 2 event.

Bit 22 **T**IM**CCMP1**: Timer C compare 1
The timer B counter is reset upon timer C compare 1 event.

Bit 21 **T**IMA**CMP4**: Timer A compare 4
The timer B counter is reset upon timer A compare 4 event.

Bit 20 **T**IMA**CMP2**: Timer A compare 2
The timer B counter is reset upon timer A compare 2 event.

Bit 19 **T**IMA**CMP1**: Timer A compare 1
The timer B counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event
The timer B counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9
The timer B counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer B counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer B counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer B counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer B counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer B counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer B counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer B counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer B counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer B counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer B counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer B counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer B counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer B counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer B compare 4 reset
The timer B counter is reset upon timer B compare 4 event.

Bit 2 **CMP2**: Timer B compare 2 reset
The timer B counter is reset upon timer B compare 2 event.

Bit 1 **UPDT**: Timer B update reset
The timer B counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer B counter is reset upon timer F compare 1 event.

34.5.36 HRTIM timer C reset register (HRTIM_RSTCR)

Address offset: 0x1D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIME CMP4	TIME CMP2	TIME CMP1	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2
The timer C counter is reset upon timer F Compare 2 event.

Bit 30 TIME**CPM4**: Timer E compare 4
The timer C counter is reset upon timer E compare 4 event.

Bit 29 **T**IM**ECMP2**: Timer E compare 2
The timer C counter is reset upon timer E compare 2 event.

Bit 28 **T**IM**ECMP1**: Timer E compare 1
The timer C counter is reset upon timer E compare 1 event.

Bit 27 **T**IM**DCMP4**: Timer D compare 4
The timer C counter is reset upon timer D compare 4 event.

Bit 26 **T**IM**DCMP2**: Timer D compare 2
The timer C counter is reset upon timer D compare 2 event.

Bit 25 **T**IM**DCMP1**: Timer D compare 1
The timer C counter is reset upon timer D compare 1 event.

Bit 24 **T**IM**BCMP4**: Timer B compare 4
The timer C counter is reset upon timer B compare 4 event.

Bit 23 **T**IM**BCMP2**: Timer B compare 2
The timer C counter is reset upon timer B compare 2 event.

Bit 22 **T**IM**BCMP1**: Timer B compare 1
The timer C counter is reset upon timer B compare 1 event.

Bit 21 **T**IMA**CMP4**: Timer A compare 4
The timer C counter is reset upon timer A compare 4 event.

Bit 20 **T**IMA**CMP2**: Timer A compare 2
The timer C counter is reset upon timer A compare 2 event.

Bit 19 **T**IMA**CMP1**: Timer A compare 1
The timer C counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event
The timer C counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9
The timer C counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer C counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer C counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer C counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer C counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer C counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer C counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer C counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer C counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer C counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer C counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer C counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer C counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer C counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer C compare 4 reset
The timer C counter is reset upon timer C compare 4 event.

Bit 2 **CMP2**: Timer C compare 2 reset
The timer C counter is reset upon timer C compare 2 event.

Bit 1 **UPDT**: Timer C update reset
The timer C counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer C counter is reset upon timer F compare 1 event.

34.5.37 HRTIM timer D reset register (HRTIM_RSTDR)

Address offset: 0x254

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIME CMP4	TIME CMP2	TIME CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2
The timer D counter is reset upon timer F Compare 2 event.

Bit 30 TIME**CPM4**: Timer E compare 4
The timer D counter is reset upon timer E compare 4 event.

Bit 29 **T**IM**ECMP2**: Timer E compare 2
The timer D counter is reset upon timer E compare 2 event.

Bit 28 **T**IM**ECMP1**: Timer E compare 1
The timer D counter is reset upon timer E compare 1 event.

Bit 27 **T**IM**CCMP4**: Timer C compare 4
The timer D counter is reset upon timer C compare 4 event.

Bit 26 **T**IM**CCMP2**: Timer C compare 2
The timer D counter is reset upon timer C compare 2 event.

Bit 25 **T**IM**CCMP1**: Timer C compare 1
The timer D counter is reset upon timer C compare 1 event.

Bit 24 **T**IM**BCMP4**: Timer B compare 4
The timer D counter is reset upon timer B compare 4 event.

Bit 23 **T**IM**BCMP2**: Timer B compare 2
The timer D counter is reset upon timer B compare 2 event.

Bit 22 **T**IM**BCMP1**: Timer B compare 1
The timer D counter is reset upon timer B compare 1 event.

Bit 21 **T**IMA**CMP4**: Timer A compare 4
The timer D counter is reset upon timer A compare 4 event.

Bit 20 **T**IMA**CMP2**: Timer A compare 2
The timer D counter is reset upon timer A compare 2 event.

Bit 19 **T**IMA**CMP1**: Timer A compare 1
The timer D counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event
The timer D counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9
The timer D counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer D counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer D counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer D counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer D counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer D counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer D counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer D counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer D counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer D counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer D counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer D counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer D counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer D counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer D compare 4 reset
The timer D counter is reset upon timer D compare 4 event.

Bit 2 **CMP2**: Timer D compare 2 reset
The timer D counter is reset upon timer D compare 2 event.

Bit 1 **UPDT**: Timer D update reset
The timer D counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer D counter is reset upon timer F compare 1 event.

34.5.38 HRTIM timer E reset register (HRTIM_RSTER)

Address offset: 0x2D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMF CMP2	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIMF CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCPM2**: Timer F compare 2
The timer E counter is reset upon timer F Compare 2 event.

Bit 30 TIMD**CPM4**: Timer D compare 4
The timer E counter is reset upon timer D compare 4 event.

Bit 29 **T**IM**DCMP2**: Timer D compare 2
The timer E counter is reset upon timer D compare 2 event.

Bit 28 **T**IM**DCMP1**: Timer D compare 1
The timer E counter is reset upon timer D compare 1 event.

Bit 27 **T**IM**CCMP4**: Timer C compare 4
The timer E counter is reset upon timer C compare 4 event.

Bit 26 **T**IM**CCMP2**: Timer C compare 2
The timer E counter is reset upon timer C compare 2 event.

Bit 25 **T**IM**CCMP1**: Timer C compare 1
The timer E counter is reset upon timer C compare 1 event.

Bit 24 **T**IM**BCMP4**: Timer B compare 4
The timer E counter is reset upon timer B compare 4 event.

Bit 23 **T**IM**BCMP2**: Timer B compare 2
The timer E counter is reset upon timer B compare 2 event.

Bit 22 **T**IMB**CMP1**: Timer B compare 1
The timer E counter is reset upon timer B compare 1 event.

Bit 21 **T**IMA**CMP4**: Timer A compare 4
The timer E counter is reset upon timer A compare 4 event.

Bit 20 **T**IMA**CMP2**: Timer A compare 2
The timer E counter is reset upon timer A compare 2 event.

Bit 19 **T**IMA**CMP1**: Timer A compare 1
The timer E counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event
The timer E counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9
The timer E counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer E counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer E counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer E counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer E counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer E counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer E counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer E counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer E counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer E counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer E counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer E counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer E counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer E counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer E compare 4 reset
The timer E counter is reset upon timer E compare 4 event.

Bit 2 **CMP2**: Timer E compare 2 reset
The timer E counter is reset upon timer E compare 2 event.

Bit 1 **UPDT**: Timer E update reset
The timer E counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer E counter is reset upon timer F compare 1 event.

34.5.39 HRTIM timer F reset register (HRTIM_RSTFR)

Address offset: 0x354

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME CMP2	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXT EVNT 10	EXT EVNT9	EXT EVNT8
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXT EVNT7	EXT EVNT6	EXT EVNT5	EXT EVNT4	EXT EVNT3	EXT EVNT2	EXT EVNT1	MST CMP4	MST CMP3	MST CMP2	MST CMP1	MST PER	CMP4	CMP2	UPDT	TIME CMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMECPM2**: Timer E compare 2
The timer F counter is reset upon timer E Compare 2 event.

Bit 30 TIMDC**PM4**: Timer D compare 4
The timer F counter is reset upon timer D compare 4 event.

Bit 29 **T**IM**DCMP2**: Timer D compare 2
The timer F counter is reset upon timer D compare 2 event.

Bit 28 **T**IMD**CMP1**: Timer D compare 1
The timer F counter is reset upon timer D compare 1 event.

Bit 27 **T**IM**CCMP4**: Timer C compare 4
The timer F counter is reset upon timer C compare 4 event.

Bit 26 **T**IMC**CMP2**: Timer C compare 2
The timer F counter is reset upon timer C compare 2 event.

Bit 25 **T**IMC**CMP1**: Timer C compare 1
The timer F counter is reset upon timer C compare 1 event.

Bit 24 **T**IMB**CMP4**: Timer B compare 4
The timer F counter is reset upon timer B compare 4 event.

Bit 23 **T**IMB**CMP2**: Timer B compare 2
The timer F counter is reset upon timer B compare 2 event.

Bit 22 **T**IMB**CMP1**: Timer B compare 1
The timer F counter is reset upon timer B compare 1 event.

Bit 21 **T**IMA**CMP4**: Timer A compare 4
The timer F counter is reset upon timer A compare 4 event.

Bit 20 **T**IMA**CMP2**: Timer A compare 2
The timer F counter is reset upon timer A compare 2 event.

Bit 19 **T**IMA**CMP1**: Timer A compare 1
The timer F counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNT10**: External event
The timer F counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: External event 9
The timer F counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: External event 8
The timer F counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: External event 7
The timer F counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: External event 6
The timer F counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: External event 5
The timer F counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: External event 4
The timer F counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: External event 3
The timer F counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: External event 2
The timer F counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: External event 1
The timer F counter is reset upon external event 1.

Bit 8 **MSTCMP4**: Master compare 4
The timer F counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer F counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer F counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer F counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER** Master timer period
The timer F counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer F compare 4 reset
The timer F counter is reset upon timer F compare 4 event.

Bit 2 **CMP2**: Timer F compare 2 reset
The timer F counter is reset upon timer F compare 2 event.

Bit 1 **UPDT**: Timer F update reset
The timer F counter is reset upon update event.

Bit 0 **TIMECMP1**: Timer E compare 1
The timer F counter is reset upon timer E compare 1 event.

34.5.40 HRTIM timer x chopper register (HRTIM_CHPxR) (x = A to F)

Address offset: Block A: 0x0D8

Address offset: Block B: 0x158

Address offset: Block C: 0x1D8

Address offset: Block D: 0x258

Address offset: Block E: 0x2D8

Address offset: Block F: 0x358

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	STRTPW[3:0]				CARDTY[2:0]			CARFRQ[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:7 **STRPW[3:0]**: Timer x start pulsewidth

This register defines the initial pulsewidth following a rising edge on output signal.
This bit field cannot be modified when one of the CHPx bits is set.
t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.
0000: 53.3 ns (1/18.75 MHz)
...
1111: 853.3 ns (16/18.75 MHz)

Bits 6:4 **CARDTY[2:0]**: Timer x chopper duty cycle value

This register defines the duty cycle of the carrier signal. This bit field cannot be modified when one of the CHPx bits is set.
000: 0/8 (that is only 1st pulse is present)
...
111: 7/8

Bits 3:0 **CARFRQ[3:0]**: Timer x carrier frequency value

This register defines the carrier frequency F_{CHPFRQ} = f_{HRTIM} / (16 x (CARFRQ[3:0]+1)).
This bit field cannot be modified when one of the CHPx bits is set.
0000: 18.75 MHz (f_{HRTIM}/16)
...
1111: 1.172 MHz (f_{HRTIM} /256)

34.5.41 HRTIM timer A capture 1 control register (HRTIM_CPT1ACR)

Address offset: 0x0DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TF CMP2	TF CMP1	TF1 RST	TF1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

34.5.42 HRTIM timer B capture 1 control register (HRTIM_CPT1BCR)

Address offset: 0x15C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TF CMP2	TF CMP1	TF1 RST	TF1 SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

34.5.43 HRTIM timer C capture 1 control register (HRTIM_CPT1CCR)

Address offset: 0x1DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TF CMP2	TF CMP1	TF1 RST	TF1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

34.5.44 HRTIM timer D capture 1 control register (HRTIM_CPT1DCR)

Address offset: 0x25C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TF CMP2	TF CMP1	TF1 RST	TF1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

34.5.45 HRTIM timer E capture 1 control register (HRTIM_CPT1ECR)

Address offset: 0x2DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TF CMP2	TF CMP1	TF1 RST	TF1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

34.5.46 HRTIM timer F capture 1 control register (HRTIM_CPT1FCR)

Address offset: 0x35C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TE CMP2	TE CMP1	TE1 RST	TE1 SET	TD CMP2	TD CMP1	TD1 RST	TD1 SET	TC CMP2	TC CMP1	TC1 RST	TC1 SET	TB CMP2	TB CMP1	TB1 RST	TB1 SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA CMP2	TA CMP1	TA1 RST	TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

34.5.47 HRTIM timer x capture 2 control register (HRTIM_CPT2xCR) (x = A to F)

Address offset: Block A: 0x0E0

Address offset: Block B: 0x160

Address offset: Block C: 0x1E0

Address offset: Block D: 0x260

Address offset: Block E: 0x2E0

Address offset: Block F: 0x360

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TF CMP2 / TE CMP2	TF CMP1 / TE CMP1	TF1 RST / TE1 RST	TF1 SET / TE1 SET	TF CMP2 / TD CMP2	TF CMP1 / TD CMP1	TF1 RST / TD1 RST	TF1 SET / TD1 SET	TF CMP2 / TC CMP2	TF CMP1 / TC CMP1	TF1 RST / TC1 RST	TF1 SET / TC1 SET	TF CMP2 / TB CMP2	TF CMP1 / TB CMP1	TF1 RST / TB1 RST	TF1 SET / TB1 SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TF CMP2 / TA CMP2	TF CMP1 / TA CMP1	TF1 RST / TA1 RST	TF1 SET / TA1 SET	EXEV10CPT	EXEV9 CPT	EXEV8 CPT	EXEV7 CPT	EXEV6 CPT	EXEV5 CPT	EXEV4 CPT	EXEV3 CPT	EXEV2 CPT	EXEV1 CPT	UPD CPT	SW CPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2FCR:

TECMP2: Timer E compare 2

Refer to TACMP2 description.

In HRTIM_CPT2ECR:

TFCMP2: Timer F compare 2

Refer to TACMP2 description.

Bit 30 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2FCR:

TECMP1: Timer E compare 1

Refer to TACMP1 description.

In HRTIM_CPT2ECR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 29 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2FCR:

TE1RST: Timer E output 1 reset

Refer to TA1RST description.

In HRTIM_CPT2ECR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 28 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2FCR:

TE1SET: Timer E output 1 set

Refer to TA1SET description.

In HRTIM_CPT2ECR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 27 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TDCMP2: Timer D compare 2

Refer to TACMP2 description.

In HRTIM_CPT2DCR:

TFCMP2: Timer F compare 2

Refer to TACMP2 description.

Bit 26 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TDCMP1:Timer D compare 1

Refer to TACMP1 description.

In HRTIM_CPT2DCR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 25 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TD1RST: Timer D output 1 reset

Refer to TA1RST description.

In HRTIM_CPT2DCR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 24 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TD1SET: Timer D output 1 set

Refer to TA1SET description.

In HRTIM_CPT2DCR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 23 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TCCMP2: Timer C compare 2

Refer to TACMP2 description.

In HRTIM_CPT2CCR:

TFCMP2: Timer F compare 2

Refer to TACMP2 description.

Bit 22 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TCCMP1:Timer C compare 1

Refer to TACMP1 description.

In HRTIM_CPT2CCR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 21 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TC1RST: Timer C output 1 reset

Refer to TA1RST description.

In HRTIM_CPT2CCR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 20 In HRTIM_CPT2ACR, HRTIM_CPT2BCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TC1SET: Timer C output 1 set

Refer to TA1SET description.

In HRTIM_CPT2CCR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 19 In HRTIM_CPT2ACR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TBCMP2: Timer B compare 2

Refer to TACMP2 description.

In HRTIM_CPT2BCR:

TFCMP2: Timer F compare 2

Refer to TACMP2 description.

Bit 18 In HRTIM_CPT2ACR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TBCMP1: Timer B compare 1

Refer to TACMP1 description.

In HRTIM_CPT2BCR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 17 In HRTIM_CPT2ACR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TB1RST: Timer B output 1 reset

Refer to TA1RST description.

In HRTIM_CPT2BCR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 16 In HRTIM_CPT2ACR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TB1SET: Timer B output 1 set

Refer to TA1SET description.

In HRTIM_CPT2BCR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 15 In HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TACMP2: Timer A compare 2

0: No action

1: Timer A compare 2 triggers capture 2

In HRTIM_CPT2ACR:

TFCMP2: Timer E compare 2

Refer to TACMP2 description.

Bit 14 In HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TACMP1: Timer A compare 1

0: No action

1: Timer A compare 1 triggers capture 2

In HRTIM_CPT2ACR:

TFCMP1: Timer F compare 1

Refer to TACMP1 description.

Bit 13 In HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TA1RST: Timer B output 1 reset

0: No action

1: Capture 2 is triggered by TA1 output active to inactive transition

In HRTIM_CPT2ACR:

TF1RST: Timer F output 1 reset

Refer to TA1RST description.

Bit 12 In HRTIM_CPT2BCR, HRTIM_CPT2CCR, HRTIM_CPT2DCR, HRTIM_CPT2ECR, HRTIM_CPT2FCR:

TA1SET: Timer B output 1 set

0: No action

1: Capture 2 is triggered by TA1 output inactive to active transition

In HRTIM_CPT2ACR:

TF1SET: Timer F output 1 set

Refer to TA1SET description.

Bit 11 **EXEV10CPT**: External event 10 capture

Refer to EXEV1CPT description

Bit 10 **EXEV9CPT**: External event 9 capture

Refer to EXEV1CPT description.

Bit 9 **EXEV8CPT**: External event 8 capture

Refer to EXEV1CPT description.

Bit 8 **EXEV7CPT**: External event 7 capture

Refer to EXEV1CPT description.

Bit 7 **EXEV6CPT**: External event 6 capture

Refer to EXEV1CPT description.

Bit 6 **EXEV5CPT**: External event 5 capture

Refer to EXEV1CPT description.

Bit 5 **EXEV4CPT**: External event 4 capture

Refer to EXEV1CPT description.

Bit 4 **EXEV3CPT**: External event 3 capture

Refer to EXEV1CPT description.

Bit 3 **EXEV2CPT**: External event 2 capture

Refer to EXEV1CPT description.

Bit 2 **EXEV1CPT**: External event 1 capture

0: No action

1: The external event 1 triggers the capture 2

Bit 1 **UPDCPT**: Update capture

0: No action

1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture

0: No action

1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

34.5.48 HRTIM timer x output register (HRTIM_OUTxR) (x = A to F)

Address offset: Block A: 0x0E4

Address offset: Block B: 0x164

Address offset: Block C: 0x1E4

Address offset: Block D: 0x264

Address offset: Block E: 0x2E4

Address offset: Block F: 0x364

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM 2	POL2	Res.
								rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BIAR	Res.	DLYPRT[2:0]			DLYPR TEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM 1	POL1	Res.
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DIDL2**: Output 2 deadtime upon burst mode Idle entry

This bit delays the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering in idle state during a burst mode operation.

0: The programmed Idle state is applied immediately to the output 2

1: Deadtime (inactive level) is inserted on output 2 before entering the idle mode. The deadtime value is set by DTFx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 is set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 22 **CHP2**: Output 2 chopper enable

This bit enables the chopper on output 2.

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 21:20 **FAULT2[1:0]**: Output 2 fault state

These bits select the output 2 state after a fault event.

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 19 **IDLES2**: Output 2 idle state

This bit selects the output 2 idle state.

0: Inactive

1: Active

Note: This parameter must be set prior to having the HRTIM controlling the outputs.

Bit 18 **IDLEM2**: Output 2 idle mode
This bit selects the output 2 idle mode.
0: No action: the output is not affected by the burst mode operation.
1: The output is in idle state when requested by the burst mode controller.
Note: This bit is preloaded and is changed during run-time, but must not be changed while the burst mode is active.

Bit 17 **POL2**: Output 2 polarity
This bit selects the output 2 polarity.
0: Positive polarity (output active high)
1: Negative polarity (output active low)
Note: This parameter cannot be changed once the timer x is enabled.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **BIAR**: Balanced Idle Automatic Resume
This bit selects is the outputs are automatically re-enabled after a balanced idle event.
This bit is only significant if DLYPRT[2:0] = 011 or 111, it is ignored otherwise.
0: Disabled
1: Enabled
Note: This parameter cannot be changed once the timer x is enabled.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DLYPRT[2:0]**: Delayed protection
These bits define the source and outputs on which the delayed protection schemes are applied.
In HRTIM_OUTAR, HRTIM_OUTBR, HRTIM_OUTCR:
000: Output 1 delayed idle on external event 6
001: Output 2 delayed idle on external event 6
010: Output 1 and output 2 delayed idle on external event 6
011: Balanced idle on external event 6
100: Output 1 delayed idle on external event 7
101: Output 2 delayed idle on external event 7
110: Output 1 and output 2 delayed idle on external event 7
111: Balanced idle on external event 7
In HRTIM_OUTDR, HRTIM_OUTER, HRTIM_OUTFR:
000: Output 1 delayed idle on external event 8
001: Output 2 delayed idle on external event 8
010: Output 1 and output 2 delayed idle on external event 8
011: Balanced idle on external event 8
100: Output 1 delayed idle on external event 9
101: Output 2 delayed idle on external event 9
110: Output 1 and output 2 delayed idle on external event 9
111: Balanced idle on external event 9
Note: This bit field must not be modified once the delayed protection is enabled (DLYPRTEN bit set).

Bit 9 **DLYPRTEN**: Delayed protection enable
This bit enables the delayed protection scheme.
0: No action
1: Delayed protection is enabled, as per DLYPRT[2:0] bits
Note: This parameter cannot be changed once the timer x is enabled (TxEN bit set).

Bit 8 **DTEN**: Deadtime enable

This bit enables the deadtime insertion on output 1 and output 2.

0: Output 1 and output 2 signals are independent.

1: Deadtime is inserted between output 1 and output 2 (reference signal is output 1 signal generator)

Note: This parameter cannot be changed once the timer is operating (TxEN bit set) or if its outputs are enabled and set/reset by another timer.

Bit 7 **DIDL1**: Output 1 deadtime upon burst mode idle entry

This bit delays the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering the idle state during a burst mode operation.

0: The programmed idle state is applied immediately to the output 1

1: Deadtime (inactive level) is inserted on output 1 before entering the idle mode. The deadtime value is set by DTRx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 is set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 6 **CHP1**: Output 1 chopper enable

This bit enables the chopper on output 1.

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 5:4 **FAULT1[1:0]**: Output 1 fault state

These bits select the output 1 state after a fault event

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 3 **IDLES1**: Output 1 Idle State

This bit selects the output 1 idle state.

0: Inactive

1: Active

Note: This parameter must be set prior to HRTIM controlling the outputs.

Bit 2 **IDLEM1**: Output 1 Idle mode

This bit selects the output 1 idle mode.

0: No action: the output is not affected by the burst mode operation.

1: The output is in idle state when requested by the burst mode controller.

Note: This bit is preloaded and is changed during runtime, but must not be changed while burst mode is active.

Bit 1 **POL1**: Output 1 polarity

This bit selects the output 1 polarity.

0: Positive polarity (output active high)

1: Negative polarity (output active low)

Note: This parameter cannot be changed once the timer x is enabled.

Bit 0 Reserved, must be kept at reset value.

34.5.49 HRTIM timer x fault register (HRTIM_FLTxR) (x = A to F)

Address offset: Block A: 0x0E8

Address offset: Block B: 0x168

Address offset: Block C: 0x1E8

Address offset: Block D: 0x268

Address offset: Block E: 0x2E8

Address offset: Block F: 0x368

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT LCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rwo															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT6 EN	FLT5 EN	FLT4 EN	FLT3 EN	FLT2 EN	FLT1 EN
										rw	rw	rw	rw	rw	rw

Bit 31 **FLTLCK**: Fault sources lock
0: FLT1EN..FLT6EN bits are read/write
1: FLT1EN..FLT6EN bits are read only
The FLTLCK bit is write-once. Once it has been set, it cannot be modified till the next system or IP reset.

Bits 30:6 Reserved, must be kept at reset value.

Bit 5 **FLT6EN**: Fault 6 enable
0: Fault 6 input ignored
1: Fault 6 input is active and disables HRTIM outputs

Bit 4 **FLT5EN**: Fault 5 enable
0: Fault 5 input ignored
1: Fault 5 input is active and disables HRTIM outputs

Bit 3 **FLT4EN**: Fault 4 enable
0: Fault 4 input ignored
1: Fault 4 input is active and disables HRTIM outputs

Bit 2 **FLT3EN**: Fault 3 enable
0: Fault 3 input ignored
1: Fault 3 input is active and disables HRTIM outputs

Bit 1 **FLT2EN**: Fault 2 enable
0: Fault 2 input ignored
1: Fault 2 input is active and disables HRTIM outputs

Bit 0 **FLT1EN**: Fault 1 enable
0: Fault 1 input ignored
1: Fault 1 input is active and disables HRTIM outputs

34.5.50 HRTIM timer x control register 2 (HRTIM_TIMxCR2) (x = A to F)

Address offset: Block A: 0x0EC

Address offset: Block B: 0x16C

Address offset: Block C: 0x1EC

Address offset: Block D: 0x26C

Address offset: Block E: 0x2EC

Address offset: Block F: 0x36C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRG HLF	Res.	Res.	GT CMP3	GT CMP1
											rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEROM[1:0]		BMROM[1:0]		ADROM[1:0]		OUTROM[1:0]		ROM[1:0]		Res.	UDM	Res.	DCDR	DCDS	DCDE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 20 **TRGHLF**: Triggered-half mode

This bit field defines whether the compare 2 register is behaving in standard mode (compare match issued as soon as counter equal compare), or in triggered-half mode (refer to *Triggered-half mode*).

0: CMP2 register is written by the user only (standard compare mode)

1: CMP2 value is set by hardware as soon as a capture 1 event occurs. It is loaded with the (capture 1 divided by 2) value. The initial value can be written by the user (as long as TRGHLF is reset), but is ignored once the first capture has been triggered (the preload mechanism is disabled for CMP2 when the TRGHLF bit is set).

Note: This bit field must not be modified once the counter is enabled (TxCEN bit set).

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **GTCMP3**: Greater than compare 3 PWM mode

This bit defines the compare 3 operating mode.

0: The compare 3 event is generated when the counter is equal to the compare value (compare match mode)

1: The compare 3 event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

Bit 16 **GTCMP1**: Greater than compare 1 PWM mode

This bit defines the compare 1 operating mode.

0: The compare 1 event is generated when the counter is equal to the compare value (compare match mode)

1: The compare 1 event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

Bits 15:14 **FEROM[1:0]**: Fault and event roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the Roll-over event used by the fault and event counters.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bit field cannot be changed once the timer is operating (TxEN bit set).

Bits 13:12 **BMROM[1:0]**: Burst mode roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event used in the burst mode controller, as clock as burst mode trigger.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This parameter cannot be changed once the timer is operating (TxEN bit set).

Bits 11:10 **ADROM[1:0]**: ADC roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event which triggers the ADC.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bit field cannot be changed once the timer is operating (TxEN bit set).

Bits 9:8 **OUTROM[1:0]**: Output roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event which sets and/or resets the ouputs, as per HRTIM_SETxyR and HRTIM_RSTxyR settings.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bit field cannot be changed once the timer is operating (TxEN bit set).

Bits 7:6 **ROM[1:0]**: Roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event with the following destinations: update trigger (to transfer content from preload to active registers), IRQ and DMA requests, repetition counter decrement and external event filtering.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bit field cannot be changed once the timer is operating (TxEN bit set).

Bit 5 Reserved, must be kept at reset value.

Bit 4 **UDM**: Up-Down mode
This bit defines if the counter is operating in up or up-down counting mode.
0: The counter is operating in up-counting mode
1: The counter is operating in up-down counting mode
Note: This bit cannot be changed once the timer is operating (TxEN bit set).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCDR**: Dual channel DAC reset trigger
This bit defines when the hrtim_dac_reset_trgx trigger is generated.
0: The trigger is generated on counter reset or roll-over event
1: The trigger is generated on output 1 set event
Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

Bit 1 **DCDS** Dual channel DAC Step trigger
This bit defines when the hrtim_dac_step_trgx trigger is generated.
0: The trigger is generated on compare 2 event
1: The trigger is generated on output 1 reset event
Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

Bit 0 **DCDE**: Dual channel DAC trigger enable
This bit enables the dual channel DAC triggering mechanism.
0: Dual channel DAC trigger disabled
1: Dual channel DAC trigger enabled
Note: This bit cannot be changed once the timer is operating (TxEN bit set).

34.5.51 HRTIM timer x external event filtering register 3 (HRTIM_EEFxR3) (x = A to F)

Address offset: Block A: 0x0F0

Address offset: Block B: 0x170

Address offset: Block C: 0x1F0

Address offset: Block D: 0x270

Address offset: Block E: 0x2F0

Address offset: Block F: 0x370

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EEVACNT[5:0]						EEVASEL[3:0]				Res.	EEVARSTM	EEVACRES	EEVACE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **EEVACNT[5:0]** External event A counter

This bit field selects the external event A counter threshold. An event is considered valid when the number of events is equal to the (EEVACNT[5:0]+1) value.

Bits 7:4 **EEVASEL[3:0]**: External event A Selection

This bit selects the external event A source.
0: External event 1 is used as external event A source
1: External event 2 is used as external event A source
...
9: External event 10 is used as external event A source
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EEVARSTM**: External event A reset mode

This bit selects the external event x counter reset mode.
0: External event counter A is reset on each reset / roll-over event
1: External event counter A is reset on each reset / roll-over event only if no event occurs during last counting period

Bit 1 **EEVACRES**: External event A counter reset

This bit resets the external event A counter. It is set by software and reset by hardware.
0: No action
1: External event counter A is reset

Bit 0 **EEVACE**: External event A counter enable

This bit enables the external event x counter.
0: External event A counter disabled
1: External event A counter enabled

34.5.52 HRTIM control register 1 (HRTIM_CR1)

Address offset: 0x380

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]		
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF UDIS	TE UDIS	TD UDIS	TC UDIS	TB UDIS	TA UDIS	MUDIS
									rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:25 **AD4USRC[2:0]**: ADC trigger 4 update source

Refer to AD1USRC[2:0] description.

Bits 24:22 **AD3USRC[2:0]**: ADC trigger 3 update source

Refer to AD1USRC[2:0] description.

Bits 21:19 **AD2USRC[2:0]**: ADC trigger 2 update source

Refer to AD1USRC[2:0] description.

Bits 18:16 **AD1USRC[2:0]**: ADC trigger 1 update source

These bits define the source which triggers the update of the HRTIM_ADC1R register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, in HRTIM_MCR or HRTIM_TIMxCR.

000: Master timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110: Timer F
111: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TFUDIS**: Timer F update disable

Refer to TAUDIS description.

Bit 5 **TEUDIS**: Timer E update disable

Refer to TAUDIS description

Bit 4 **TDUDIS**: Timer D update disable

Refer to TAUDIS description.

Bit 3 **TCUDIS**: Timer C update disable

Refer to TAUDIS description.

Bit 2 **TBUDIS**: Timer B update disable

Refer to TAUDIS description.

Bit 1 **TAUDIS**: Timer A update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily on timer A.

0: Update enabled. The update occurs upon generation of the selected source.
1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 0 **MUDIS**: Master update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily.

0: Update enabled.
1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

34.5.53 HRTIM control register 2 (HRTIM_CR2)

Address offset: 0x384

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWPF	SWPE	SWPD	SWPC	SWPB	SWPA
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TFRST	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	TF SWU	TE SWU	TD SWU	TC SWU	TB SWU	TA SWU	MSWU
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **SWPF**: Swap timer F outputs
Refer to SWPA description.
Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 20 **SWPE**: Swap timer E outputs
Refer to SWPA description.
Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 19 **SWPD**: Swap timer D outputs
Refer to SWPA description.
Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 18 **SWPC**: Swap timer C outputs
Refer to SWPA description.
Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 17 **SWPB**: Swap timer B outputs
Refer to SWPA description.
Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 16 **SWPA**: Swap timer A outputs
This bit allows to swap the timer A outputs.
0: HRTIM_SETA1R and HRTIM_RSTA1R are coding for the output A1, HRTIM_SETA2R and HRTIM_RSTA2R are coding for the output A2
1: HRTIM_SETA1R and HRTIM_RSTA1R are coding for the output A2, HRTIM_SETA2R and HRTIM_RSTA2R are coding for the output A1
Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TFRST**: Timer F counter software reset
Refer to TARST description.

Bit 13 **TERST**: Timer E counter software reset
Refer to TARST description.

Bit 12 **TDRST**: Timer D counter software reset
Refer to TARST description.

Bit 11 **TCRST**: Timer C counter software reset
Refer to TARST description.

Bit 10 **TBRST**: Timer B counter software reset
Refer to TARST description.

Bit 9 **TARST**: Timer A counter software reset
Setting this bit resets the timer A counter.
The bit is automatically reset by hardware.

Bit 8 **MRST**: Master counter software reset
Setting this bit resets the master timer counter.
The bit is automatically reset by hardware.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TFSWU**: Timer F software update
Refer to TASWU description.

Bit 5 **TESWU**: Timer E software update
Refer to TASWU description.

Bit 4 **TDSWU**: Timer D software update
Refer to TASWU description.

Bit 3 **TCSWU**: Timer C software update
Refer to TASWU description.

Bit 2 **TBSWU**: Timer B software update
Refer to TASWU description.

Bit 1 **TASWU**: Timer A software update
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register and any pending update request is canceled.

Bit 0 **MSWU**: Master timer software update
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register in the master timer and any pending update request is canceled.

34.5.54 HRTIM interrupt status register (HRTIM_ISR)

Address offset: 0x388

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	DLLRDY
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT6	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
									r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPER**: Burst mode period interrupt flag
This bit is set by hardware when a single-shot burst mode operation is completed or at the end of a burst mode period in continuous mode. It is cleared by software writing it at 1.
0: No burst mode period interrupt occurred
1: Burst mode period interrupt occurred

Bit 16 **DLLRDY**: DLL Ready Interrupt Flag
This bit is set by hardware when the DLL calibration is completed. It is cleared by software writing it at 1.
0: No DLL calibration ready interrupt occurred
1: DLL calibration ready interrupt occurred

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **FLT6**: Fault 6 interrupt flag
This bit is set by hardware when fault 6 event occurs. It is cleared by software writing it at 1.
0: No fault 6 interrupt occurred
1: Fault 6 interrupt occurred

Bit 5 **SYSFLT**: System fault interrupt flag
This bit is set by hardware when system fault event occurs. It is cleared by software writing it at 1.
0: No system fault interrupt occurred
1: System fault interrupt occurred

Bit 4 **FLT5**: Fault 5 interrupt flag
This bit is set by hardware when fault 5 event occurs. It is cleared by software writing it at 1.
0: No fault 5 interrupt occurred
1: Fault 5 interrupt occurred

Bit 3 **FLT4**: Fault 4 interrupt flag
This bit is set by hardware when fault 4 event occurs. It is cleared by software writing it at 1.
0: No fault 4 interrupt occurred
1: Fault 4 interrupt occurred

Bit 2 **FLT3**: Fault 3 interrupt flag
This bit is set by hardware when fault 3 event occurs. It is cleared by software writing it at 1.
0: No fault 3 interrupt occurred
1: Fault 3 interrupt occurred

Bit 1 **FLT2**: Fault 2 interrupt flag
This bit is set by hardware when fault 2 event occurs. It is cleared by software writing it at 1.
0: No fault 2 interrupt occurred
1: Fault 2 interrupt occurred

Bit 0 **FLT1**: Fault 1 interrupt flag
This bit is set by hardware when fault 1 event occurs. It is cleared by software writing it at 1.
0: No fault 1 interrupt occurred
1: Fault 1 interrupt occurred

34.5.55 HRTIM interrupt clear register (HRTIM_ICR)

Address offset: 0x38C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	DLLRDYC
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT6C	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
									w	w	w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPERC**: Burst mode period flag clear
Writing 1 to this bit clears the BMPER flag in HRTIM_ISR register.

Bit 16 **DLLRDYC**: DLL Ready Interrupt flag Clear
Writing 1 to this bit clears the DLLRDY flag in HRTIM_ISR register.

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **FLT6C**: Fault 6 interrupt flag clear
Writing 1 to this bit clears the FLT6 flag in HRTIM_ISR register.

Bit 5 **SYSFLTC**: System fault interrupt flag clear
Writing 1 to this bit clears the SYSFLT flag in HRTIM_ISR register.

Bit 4 **FLT5C**: Fault 5 interrupt flag clear
Writing 1 to this bit clears the FLT5 flag in HRTIM_ISR register.

Bit 3 **FLT4C**: Fault 4 interrupt flag clear
Writing 1 to this bit clears the FLT4 flag in HRTIM_ISR register.

Bit 2 **FLT3C**: Fault 3 interrupt flag clear
Writing 1 to this bit clears the FLT3 flag in HRTIM_ISR register.

Bit 1 **FLT2C**: Fault 2 interrupt flag clear
Writing 1 to this bit clears the FLT2 flag in HRTIM_ISR register.

Bit 0 **FLT1C**: Fault 1 interrupt flag clear
Writing 1 to this bit clears the FLT1 flag in HRTIM_ISR register.

34.5.56 HRTIM interrupt enable register (HRTIM_IER)

Address offset: 0x390

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	DLLRDYIE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT6IE	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPERIE**: Burst mode period interrupt enable
This bit is set and cleared by software to enable/disable the burst mode period interrupt.
0: Burst mode period interrupt disabled
1: Burst mode period interrupt enabled

Bit 16 **DLLRDYIE**: DLL Ready Interrupt Enable
This bit is set and cleared by software to enable/disable the DLL ready interrupt.
0: DLL ready interrupt disabled
1: DLL ready interrupt enabled

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **FLT6IE**: Fault 6 interrupt enable
This bit is set and cleared by software to enable/disable the fault 6 interrupt.
0: Fault 6 interrupt disabled
1: Fault 6 interrupt enabled

Bit 5 **SYSFLTIE**: System fault interrupt enable
This bit is set and cleared by software to enable/disable the system fault interrupt.
0: System fault interrupt disabled
1: System fault interrupt enabled

Bit 4 **FLT5IE**: Fault 5 interrupt enable
This bit is set and cleared by software to enable/disable the fault 5 interrupt.
0: Fault 5 interrupt disabled
1: Fault 5 interrupt enabled

Bit 3 **FLT4IE**: Fault 4 interrupt enable
This bit is set and cleared by software to enable/disable the fault 4 interrupt.
0: Fault 4 interrupt disabled
1: Fault 4 interrupt enabled

Bit 2 **FLT3IE**: Fault 3 interrupt enable
This bit is set and cleared by software to enable/disable the fault 3 interrupt.
0: Fault 3 interrupt disabled
1: Fault 3 interrupt enabled

Bit 1 **FLT2IE**: Fault 2 interrupt enable
This bit is set and cleared by software to enable/disable the fault 2 interrupt.
0: Fault 2 interrupt disabled
1: Fault 2 interrupt enabled

Bit 0 **FLT1IE**: Fault 1 interrupt enable
This bit is set and cleared by software to enable/disable the fault 1 interrupt.
0: Fault 1 interrupt disabled
1: Fault 1 interrupt enabled

34.5.57 HRTIM output enable register (HRTIM_OENR)

Address offset: 0x394

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TF2OEN	TF1OEN	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
				rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2OEN**: Timer F output 2 enable
Refer to TA1OEN description.

Bit 10 **TF1OEN**: Timer F output 1 enable
Refer to TA1OEN description

Bit 9 **TE2OEN**: Timer E output 2 enable
Refer to TA1OEN description.

Bit 8 **TE1OEN**: Timer E output 1 enable
Refer to TA1OEN description.

Bit 7 **TD2OEN**: Timer D output 2 enable
Refer to TA1OEN description.

Bit 6 **TD1OEN**: Timer D output 1 enable
Refer to TA1OEN description.

Bit 5 **TC2OEN**: Timer C output 2 enable
Refer to TA1OEN description.

Bit 4 **TC1OEN**: Timer C output 1 enable
Refer to TA1OEN description.

Bit 3 **TB2OEN**: Timer B output 2 enable
Refer to TA1OEN description.

Bit 2 **TB1OEN**: Timer B output 1 enable
Refer to TA1OEN description.

Bit 1 **TA2OEN**: Timer A output 2 enable
Refer to TA1OEN description.

Bit 0 **TA1OEN**: Timer A output 1 enable
Setting this bit enables the timer A output 1. Writing “0” has no effect.
Reading the bit returns the output enable/disable status.
This bit is cleared asynchronously by hardware as soon as the timer-related fault input(s) is (are) active.
0: output A1 disabled. The output is either in fault or Idle state.
1: output A1 enabled

Note: The disable status corresponds to both idle and fault states. The output disable status is given by TA1ODS bit in the HRTIM_ODSR register.

34.5.58 HRTIM output disable register (HRTIM_ODISR)

Address offset: 0x398

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TF2 ODIS	TF1 ODIS	TE2 ODIS	TE1 ODIS	TD2 ODIS	TD1 ODIS	TC2 ODIS	TC1 ODIS	TB2 ODIS	TB1 ODIS	TA2 ODIS	TA1 ODIS
				w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2ODIS**: Timer F output 2 disable
Refer to TA1ODIS description.

Bit 10 **TF1ODIS**: Timer F output 1 disable
Refer to TA1ODIS description

Bit 9 **TE2ODIS**: Timer E output 2 disable
Refer to TA1ODIS description.

Bit 8 **TE1ODIS**: Timer E output 1 disable
Refer to TA1ODIS description.

Bit 7 **TD2ODIS**: Timer D output 2 disable
Refer to TA1ODIS description.

Bit 6 **TD1ODIS**: Timer D output 1 disable
Refer to TA1ODIS description.

Bit 5 **TC2ODIS**: Timer C output 2 disable
Refer to TA1ODIS description.

Bit 4 **TC1ODIS**: Timer C output 1 disable
Refer to TA1ODIS description.

Bit 3 **TB2ODIS**: Timer B output 2 disable
Refer to TA1ODIS description.

Bit 2 **TB1ODIS**: Timer B output 1 disable
Refer to TA1ODIS description.

Bit 1 **TA2ODIS**: Timer A output 2 disable
Refer to TA1ODIS description.

Bit 0 **TA1ODIS**: Timer A output 1 disable
Setting this bit disables the timer A output 1. The output enters the idle state, either from the run state or from the fault state.
Writing “0” has no effect.

34.5.59 HRTIM output disable status register (HRTIM_ODSR)

Address offset: 0x39C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TF2ODS	TF1ODS	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2ODS**: Timer F output 2 disable status
Refer to TA1ODS description.

Bit 10 **TF1ODS**: Timer F output 1 disable status
Refer to TA1ODS description.

Bit 9 **TE2ODS**: Timer E output 2 disable status
Refer to TA1ODS description.

Bit 8 **TE1ODS**: Timer E output 1 disable status
Refer to TA1ODS description.

Bit 7 **TD2ODS**: Timer D output 2 disable status
Refer to TA1ODS description.

Bit 6 **TD1ODS**: Timer D output 1 disable status
Refer to TA1ODS description.

Bit 5 **TC2ODS**: Timer C output 2 disable status
Refer to TA1ODS description.

Bit 4 **TC1ODS**: Timer C output 1 disable status
Refer to TA1ODS description.

Bit 3 **TB2ODS**: Timer B output 2 disable status
Refer to TA1ODS description.

Bit 2 **TB1ODS**: Timer B output 1 disable status
Refer to TA1ODS description.

Bit 1 **TA2ODS**: Timer A output 2 disable status
Refer to TA1ODS description.

Bit 0 **TA1ODS**: Timer A output 1 disable status
Reading the bit returns the output disable status. It is not significant when the output is active (Tx1OEN or Tx2OEN = 1).
0: Output A1 disabled, in Idle state.
1: Output A1 disabled, in fault state.

34.5.60 HRTIM burst mode control register (HRTIM_BMCR)

Address offset: 0x3A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFBM	TEBM	TDBM	TCBM	TBBM	TABM	MTBM
rc_w0									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BMSTAT**: Burst mode status
This bit gives the current operating state.
0: Normal operation
1: Burst operation on-going. Writing this bit to 0 causes a burst mode early termination.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **TFBM**: Timer F burst mode
Refer to TABM description.

Bit 21 **TEBM**: Timer E burst mode
Refer to TABM description.

Bit 20 **TDBM**: Timer D burst mode
Refer to TABM description.

Bit 19 **TCBM**: Timer C burst mode
Refer to TABM description.

Bit 18 **TBBM**: Timer B burst mode
Refer to TABM description.

Bit 17 **TABM**: Timer A burst mode
This bit defines how the timer behaves during a burst mode operation. This bit field cannot be changed while the burst mode is enabled.
0: TA counter clock is maintained and the timer operates normally
1: TA counter clock is stopped and the counter is reset
Note: This bit must not be set when the balanced idle mode is active (DLYPRT[2:0] = 0x11).

Bit 16 **MTBM**: Master timer burst mode
This bit defines how the timer behaves during a burst mode operation. This bit field cannot be changed while the burst mode is enabled.
0: Master Timer counter clock is maintained and the timer operates normally
1: Master Timer counter clock is stopped and the counter is reset

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **BMPREN**: Burst mode preload enable
This bit enables the registers preload mechanism and defines whether a write access into a preloadable register (HRTIM_BMCMPR, HRTIM_BMPER) is done into the active or the preload register.
0: Preload disabled: the write access is directly done into active registers
1: Preload enabled: the write access is done into preload registers

Bits 9:6 **BMPRSC[3:0]**: Burst mode prescaler
Defines the prescaling ratio of the f_{HRTIM} clock for the burst mode controller. This bit field cannot be changed while the burst mode is enabled.
0000: Clock not divided
0001: Division by 2
0010: Division by 4
0011: Division by 8
0100: Division by 16
0101: Division by 32
0110: Division by 64
0111: Division by 128
1000: Division by 256
1001: Division by 512
1010: Division by 1024
1011: Division by 2048
1100: Division by 4096
1101: Division by 8192
1110: Division by 16384
1111: Division by 32768

Bits 5:2 **BMCLK[3:0]**: Burst mode clock source

This bit field defines the clock source for the burst mode counter. It cannot be changed while the burst mode is enabled (refer to *Table 381: Burst mode clock sources from general purpose timer* for on-chip events 1..4 connections details).

0000: Master timer counter reset/roll-over
0001: Timer A counter reset/roll-over
0010: Timer B counter reset/roll-over
0011: Timer C counter reset/roll-over
0100: Timer D counter reset/roll-over
0101: Timer E counter reset/roll-over
0110: On-chip event 1 (hrtim_bm_ck1), acting as a burst mode counter clock
0111: On-chip event 2 (hrtim_bm_ck2) acting as a burst mode counter clock
1000: On-chip event 3 (hrtim_bm_ck3) acting as a burst mode counter clock
1001: On-chip event 4 (hrtim_bm_ck4) acting as a burst mode counter clock
1010: Prescaled f_{HRTIM} clock (as per BMPRSC[3:0] setting)
1011: Timer F counter reset/roll-over
Others: Reserved

Bit 1 **BMOM**: Burst mode operating mode

This bit defines if the burst mode is entered once or if it is continuously operating.
0: Single-shot mode
1: Continuous operation

Bit 0 **BME**: Burst mode enable

This bit starts the burst mode controller which becomes ready to receive the start trigger. Writing this bit to 0 causes a burst mode early termination.
0: Burst mode disabled
1: Burst mode enabled

34.5.61 HRTIM burst mode trigger register (HRTIM_BMTRGR)

Address offset: 0x3A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OCHPEV	EEV8	EEV7	TDEEV8	TAEEV7	TECMP2	TECMP1	TEREP	TFCMP1	TDCMP2	TFREP	TDREP	TDRST	TFRST	TCCMP1	TCREP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OCHPEV**: On-chip event

A rising edge on the hrtim_bm_trg input (connected to general purpose timer TRGO output) triggers a burst mode entry (refer *Table 379: HRTIM inputs/outputs summary* for details).

Bit 30 **EEV8**: External event 8 (TIMD filters applied)

The external event 8 conditioned by TIMD filters is starting the burst mode operation.

Bit 29 **EEV7**: External event 7 (TIMA filters applied)

The external event 7 conditioned by TIMA filters is starting the burst mode operation.

Bit 28 **TDEEV8**: Timer D period following external event 8
The timer D period following an external event 8 (conditioned by TIMD filters) is starting the burst mode operation.

Bit 27 **TAEEV7**: Timer A period following external event 7
The timer A period following an external event 7 (conditioned by TIMA filters) is starting the burst mode operation.

Bit 26 **TECMP2**: Timer E compare 2 event
Refer to TACMP1 description.

Bit 25 **TECMP1**: Timer E compare 1 event
Refer to TACMP1 description.

Bit 24 **TEREP**: Timer E repetition
Refer to TAREP description.

Bit 23 **TFCMP1**: Timer F compare 1 event
Refer to TACMP1 description.

Bit 22 **TDCMP2**: Timer D compare 2 event
Refer to TACMP1 description.

Bit 21 **TFREP**: Timer F repetition
Refer to TAREP description.

Bit 20 **TDREP**: Timer D repetition
Refer to TAREP description.

Bit 19 **TDRST**: Timer D reset or roll-over
Refer to TARST description.

Bit 18 **TFRST**: Timer F reset
Refer to TARST description.

Bit 17 **TCCMP1**: Timer C compare 1 event
Refer to TACMP1 description.

Bit 16 **TCREP**: Timer C repetition
Refer to TAREP description.

Bit 15 **TCRST**: Timer C reset or roll-over
Refer to TARST description.

Bit 14 **TBCMP2**: Timer B compare 2 event
Refer to TACMP1 description.

Bit 13 **TBCMP1**: Timer B compare 1 event
Refer to TACMP1 description.

Bit 12 **TBREP**: Timer B repetition
Refer to TAREP description.

Bit 11 **TBRST**: Timer B reset or roll-over
Refer to TARST description.

Bit 10 **TACMP2**: Timer A compare 2 event
Refer to TACMP1 description.

Bit 9 **TACMP1**: Timer A compare 1 event
The timer A compare 1 event is starting the burst mode operation.

Bit 8 **TAREP**: Timer A repetition
The Timer A repetition event is starting the burst mode operation.

Bit 7 **TARST**: Timer A reset or roll-over
The Timer A reset or roll-over event is starting the burst mode operation.

Bit 6 **MSTCMP4**: Master compare 4
Refer to MSTCMP1 description.

Bit 5 **MSTCMP3**: Master compare 3
Refer to MSTCMP1 description.

Bit 4 **MSTCMP2**: Master compare 2
Refer to MSTCMP1 description.

Bit 3 **MSTCMP1**: Master compare 1
The master timer compare 1 event is starting the burst mode operation.

Bit 2 **MSTREP**: Master repetition
The master timer repetition event is starting the burst mode operation.

Bit 1 **MSTRST**: Master reset or roll-over
The master timer reset and roll-over event is starting the burst mode operation.

Bit 0 **SW**: Software start
This bit is set by software and automatically reset by hardware.
When set, It starts the burst mode operation immediately.
This bit is not active if the burst mode is not enabled (BME bit is reset).

34.5.62 HRTIM burst mode compare register (HRTIM_BMCMPR)

Address offset: 0x3A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BMCMP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BMCMP[15:0]**: Burst mode compare value
Defines the number of periods during which the selected timers are in idle state.
This register holds either the content of the preload register or the content of the active register if the preload is disabled.

Note: BMCMP[15:0] cannot be set to 0x0000 when using the f_{HRTIM} *clock without a prescaler as the burst mode clock source (BMCLK[3:0] = 1010 and BMPRESC[3:0] = 0000).*

34.5.63 HRTIM burst mode period register (HRTIM_BMPER)

Address offset: 0x3AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BMPER[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BMPER[15:0]**: Burst mode period

Defines the burst mode repetition period.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

Note: The BMPER[15:0] must not be null when the burst mode is enabled.

34.5.64 HRTIM timer external event control register 1 (HRTIM_EECR1)

Address offset: 0x3B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1]
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE3SNS[0]	EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **EE5FAST**: External event 5 fast mode

Refer to EE1FAST description.

Bits 28:27 **EE5SNS[1:0]**: External event 5 sensitivity

Refer to EE1SNS[1:0] description.

Bit 26 **EE5POL**: External event 5 polarity

Refer to EE1POL description.

Bits 25:24 **EE5SRC[1:0]**: External event 5 source

Refer to EE1SRC[1:0] description.

Bit 23 **EE4FAST**: External event 4 fast mode

Refer to EE1FAST description.

Bits 22:21 **EE4SNS[1:0]**: External event 4 sensitivity
Refer to EE1SNS[1:0] description.

Bit 20 **EE4POL**: External event 4 polarity
Refer to EE1POL description.

Bits 19:18 **EE4SRC[1:0]**: External event 4 source
Refer to EE1SRC[1:0] description.

Bit 17 **EE3FAST**: External event 3 fast mode
Refer to EE1FAST description.

Bits 16:15 **EE3SNS[1:0]**: External event 3 sensitivity
Refer to EE1SNS[1:0] description.

Bit 14 **EE3POL**: External event 3 polarity
Refer to EE1POL description.

Bits 13:12 **EE3SRC[1:0]**: External event 3 source
Refer to EE1SRC[1:0] description.

Bit 11 **EE2FAST**: External event 2 fast mode
Refer to EE1FAST description.

Bits 10:9 **EE2SNS[1:0]**: External event 2 sensitivity
Refer to EE1SNS[1:0] description.

Bit 8 **EE2POL**: External event 2 polarity
Refer to EE1POL description.

Bits 7:6 **EE2SRC[1:0]**: External event 2 source
Refer to EE1SRC[1:0] description.

Bit 5 **EE1FAST**: External event 1 fast mode
0: External event 1 is re-synchronized by the HRTIM logic before acting on outputs, which adds a f_{HRTIM} clock-related latency
1: External event 1 is acting asynchronously on outputs (low latency mode)
Note: This bit must not be modified once the counter in which the event is used is enabled (TxCEN bit set).

Bits 4:3 **EE1SNS[1:0]**: External event 1 sensitivity
00: On active level defined by EE1POL bit
01: Rising edge, whatever EE1POL bit value
10: Falling edge, whatever EE1POL bit value
11: Both edges, whatever EE1POL bit value

Bit 2 **EE1POL**: External event 1 polarity
This bit is only significant if EE1SNS[1:0] = 00.
0: External event is active high
1: External event is active low
Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

Bits 1:0 **EE1SRC[1:0]**: External event 1 source
This bit field selects the External event 1 source. Refer to the External events mapping and associated features table in *Chapter 5: Device configuration* for details.
00: hrtim_eev1_1
01: hrtim_eev1_2
10: hrtim_eev1_3
11: hrtim_eev1_4

Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

34.5.65 HRTIM timer external event control register 2 (HRTIM_EECR2)

Address offset: 0x3B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8SNS[0]	EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[1:0]	
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:27 **EE10SNS[1:0]**: External event 10 sensitivity
Refer to EE1SNS[1:0] description.

Bit 26 **EE10POL**: External event 10 polarity
Refer to EE1POL description.

Bits 25:24 **EE10SRC[1:0]**: External event 10 source
Refer to EE1SRC[1:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EE9SNS[1:0]**: External event 9 sensitivity
Refer to EE1SNS[1:0] description.

Bit 20 **EE9POL**: External event 9 polarity
Refer to EE1POL description.

Bits 19:18 **EE9SRC[1:0]**: External event 9 source
Refer to EE1SRC[1:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:15 **EE8SNS[1:0]**: External event 8 sensitivity
Refer to EE1SNS[1:0] description.

Bit 14 **EE8POL**: External event 8 polarity
Refer to EE1POL description.

Bits 13:12 **EE8SRC[1:0]**: External event 8 source
Refer to EE1SRC[1:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EE7SNS[1:0]**: External event 7 sensitivity
Refer to EE1SNS[1:0] description.

Bit 8 **EE7POL**: External event 7 polarity
Refer to EE1POL description.

Bits 7:6 **EE7SRC[1:0]**: External event 7 source
Refer to EE1SRC[1:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EE6SNS[1:0]**: External event 6 sensitivity
Refer to EE1SNS[1:0] description.

Bit 2 **EE6POL**: External event 6 polarity
Refer to EE1POL description.

Bits 1:0 **EE6SRC[1:0]**: External event 6 source
Refer to EE1SRC[1:0] description.

34.5.66 HRTIM timer external event control register 3 (HRTIM_EECR3)

Address offset: 0x3B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EEVSD[1:0]		Res.	Res.	EE10F[3:0]				Res.	Res.	EE9F[3:0]				Res.	Res.
rw	rw			rw	rw	rw	rw			rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8F[3:0]				Res.	Res.	EE7F[3:0]				Res.	Res.	EE6F[3:0]			
rw	rw	rw	rw			rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:30 **EEVSD[1:0]**: External event sampling clock division
This bit field indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the external event signal sampling clock (f_{EEVS}) used by the digital filters.
00: $f_{EEVS}=f_{HRTIM}$
01: $f_{EEVS}=f_{HRTIM}$ / 2
10: $f_{EEVS}=f_{HRTIM}$ / 4
11: $f_{EEVS}=f_{HRTIM}$ / 8

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **EE10F[3:0]**: External event 10 filter
Refer to EE6F[3:0] description.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:18 **EE9F[3:0]**: External event 9 filter
Refer to EE6F[3:0] description.

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:12 **EE8F[3:0]**: External event 8 filter
Refer to EE6F[3:0] description.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **EE7F[3:0]**: External event 7 filter
Refer to EE6F[3:0] description.

Bits 4:5 Reserved, must be kept at reset value.

Bits 3:0 **EE6F[3:0]**: External event 6 filter
This bit field defines the frequency used to sample external event 6 input and the length of the digital filter applied to EEV6. The digital filter is made of a counter in which N valid samples are needed to validate a transition on the output.
0000: Filter disabled
0001: $f_{SAMPLING}= f_{HRTIM}$, N=2
0010: $f_{SAMPLING}= f_{HRTIM}$, N=4
0011: $f_{SAMPLING}= f_{HRTIM}$, N=8
0100: $f_{SAMPLING}= f_{EEVS}/2$, N=6
0101: $f_{SAMPLING}= f_{EEVS}/2$, N=8
0110: $f_{SAMPLING}= f_{EEVS}/4$, N=6
0111: $f_{SAMPLING}= f_{EEVS}/4$, N=8
1000: $f_{SAMPLING}= f_{EEVS}/8$, N=6
1001: $f_{SAMPLING}= f_{EEVS}/8$, N=8
1010: $f_{SAMPLING}= f_{EEVS}/16$, N=5
1011: $f_{SAMPLING}= f_{EEVS}/16$, N=6
1100: $f_{SAMPLING}= f_{EEVS}/16$, N=8
1101: $f_{SAMPLING}= f_{EEVS}/32$, N=5
1110: $f_{SAMPLING}= f_{EEVS}/32$, N=6
1111: $f_{SAMPLING}= f_{EEVS}/32$, N=8

34.5.67 HRTIM ADC trigger 1 register (HRTIM_ADC1R)

Address offset: 0x3BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC1 TEPER	ADC1 TEC4	ADC1 TEC3	ADC1 TFRST	ADC1 TDPER	ADC1 TDC4	ADC1 TDC3	ADC1 TFPER	ADC1 TCPER	ADC1 TCC4	ADC1 TCC3	ADC1 TFC4	ADC1 TBRST	ADC1 TBPER	ADC1 TBC4	ADC1 TBC3
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC1 TFC3	ADC1 TARST	ADC1 TAPER	ADC1 TAC4	ADC1 TAC3	ADC1 TFC2	ADC1 EEV5	ADC1 EEV4	ADC1 EEV3	ADC1 EEV2	ADC1 EEV1	ADC1 MPER	ADC1 MC4	ADC1 MC3	ADC1 MC2	ADC1 MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_ADC1R bits description.

34.5.68 HRTIM ADC trigger 2 register (HRTIM_ADC2R)

Address offset: 0x3C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2 TERST	ADC2 TEC4	ADC2 TEC3	ADC2 TEC2	ADC2 TDRST	ADC2 TDPER	ADC2 TDC4	ADC2 TFPER	ADC2T DC2	ADC2T CRST	ADC2T CPER	ADC2T CC4	ADC2 TFC4	ADC2T CC2	ADC2T BPER	ADC2T BC4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC2 TFC3	ADC2 TBC2	ADC2 TAPER	ADC2 TAC4	ADC2 TFC2	ADC2 TAC2	ADC2 EEV10	ADC2 EEV9	ADC2 EEV8	ADC2 EEV7	ADC2 EEV6	ADC2 MPER	ADC2 MC4	ADC2 MC3	ADC2 MC2	ADC2 MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_ADC2R bits description.

34.5.69 HRTIM ADC trigger 3 register (HRTIM_ADC3R)

Address offset: 0x3C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC3 TEPER	ADC3 TEC4	ADC3 TEC3	ADC3 TFRST	ADC3 TDPER	ADC3 TDC4	ADC3 TDC3	ADC3 TFPER	ADC3 TCPER	ADC3 TCC4	ADC3 TCC3	ADC3 TFC4	ADC3 TBRST	ADC3 TBPER	ADC3 TBC4	ADC3 TBC3
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 TFC3	ADC3 TARST	ADC3 TAPER	ADC3 TAC4	ADC3 TAC3	ADC3 TFC2	ADC3 EEV5	ADC3 EEV4	ADC3 EEV3	ADC3 EEV2	ADC3 EEV1	ADC3 MPER	ADC3 MC4	ADC3 MC3	ADC3 MC2	ADC3 MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC3TEPER**: ADC trigger 3 on timer E period
Refer to ADC3TAPER description.

Bit 30 **ADC3TEC4**: ADC trigger 3 on timer E compare 4
Refer to ADC3TFC2 description.

Bit 29 **ADC3TEC3**: ADC trigger 3 on timer E compare 3
Refer to ADC3TFC2 description.

Bit 28 **ADC3TFRST**: ADC trigger 3 on timer F reset and counter roll-over
Refer to ADC3TARST description.

Bit 27 **ADC3TDPER**: ADC trigger 3 on timer D period
Refer to ADC3TAPER description.

Bit 26 **ADC3TDC4**: ADC trigger 3 on timer D compare 4
Refer to ADC3TFC2 description.

Bit 25 **ADC3TDC3**: ADC trigger 3 on timer D compare 3
Refer to ADC3TFC2 description.

Bit 24 **ADC3TFPER**: ADC trigger 3 on timer F period
Refer to ADC3TAPER description.

Bit 23 **ADC3TCPER**: ADC trigger 3 on timer C period
Refer to ADC3TAPER description.

Bit 22 **ADC3TCC4**: ADC trigger 3 on timer C compare 4
Refer to ADC3TFC2 description.

Bit 21 **ADC3TCC3**: ADC trigger 3 on timer C compare 3
Refer to ADC3TFC2 description.

Bit 20 **ADC3TFC4**: ADC trigger 3 on timer F compare 4
Refer to ADC3TAC2 description.

Bit 19 **ADC3TBRST**: ADC trigger 3 on timer B reset and counter roll-over
Refer to ADC3TBRST description.

Bit 18 **ADC3TBPER**: ADC trigger 3 on timer B period
Refer to ADC3TAPER description.

Bit 17 **ADC3TBC4**: ADC trigger 3 on timer B compare 4
Refer to ADC3TFC2 description.

Bit 16 **ADC3TBC3**: ADC trigger 3 on timer B compare 3
Refer to ADC3TFC2 description.

Bit 15 **ADC3TFC3**: ADC trigger 3 on timer F compare 3
Refer to ADC3TFC2 description.

Bit 14 **ADC3TARST**: ADC trigger 3 on timer A reset and counter roll-over
This bit enables the generation of an ADC trigger upon timer A reset and roll-over event, on ADC trigger 1 output.

Bit 13 **ADC3TAPER**: ADC trigger 3 on timer A period
This bit enables the generation of an ADC trigger upon timer A period event, on ADC trigger 1 output.

Bit 12 **ADC3TAC4**: ADC trigger 3 on timer A compare 4
Refer to ADC3TFC2 description.

Bit 11 **ADC3TAC3**: ADC trigger 3 on timer A compare 3
Refer to ADC3TFC2 description.

Bit 10 **ADC3TFC2**: ADC trigger 3 on timer F compare 2
This bit enables the generation of an ADC trigger upon timer F compare 2 event, on ADC trigger 3 output.

Bit 9 **ADC3EEV5**: ADC trigger 3 on external event 5
Refer to ADC3EEV1 description.

Bit 8 **ADC3EEV4**: ADC trigger 3 on external event 4
Refer to ADC3EEV1 description.

Bit 7 **ADC3EEV3**: ADC trigger 3 on external event 3
Refer to ADC3EEV1 description.

Bit 6 **ADC3EEV2**: ADC trigger 3 on external event 2
Refer to ADC3EEV1 description.

Bit 5 **ADC3EEV1**: ADC trigger 3 on external event 1
This bit enables the generation of an ADC trigger upon external event 1, on ADC trigger 3 output.

Bit 4 **ADC3MPER**: ADC trigger 3 on master period
This bit enables the generation of an ADC trigger upon master timer period event, on ADC trigger 3 output.

Bit 3 **ADC3MC4**: ADC trigger 3 on master compare 4
Refer to ADC3MC1 description.

Bit 2 **ADC3MC3**: ADC trigger 3 on master compare 3
Refer to ADC3MC1 description.

Bit 1 **ADC3MC2**: ADC trigger 3 on master compare 2
Refer to ADC3MC1 description.

Bit 0 **ADC3MC1**: ADC trigger 3 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 3 output.

34.5.70 HRTIM ADC trigger 4 register (HRTIM_ADC4R)

Address offset: 0x3C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC4 TERST	ADC4 TEC4	ADC4 TEC3	ADC4 TEC2	ADC4 TDRST	ADC4 TDPER	ADC4 TDC4	ADC4 TFPER	ADC4T DC2	ADC4T CRST	ADC4T CPER	ADC4T CC4	ADC4 TFC4	ADC4T CC2	ADC4T BPER	ADC4T BC4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC4 TFC3	ADC4 TBC2	ADC4 TAPER	ADC4 TAC4	ADC4 TFC2	ADC4 TAC2	ADC4 EEV10	ADC4 EEV9	ADC4 EEV8	ADC4 EEV7	ADC4 EEV6	ADC4 MPER	ADC4 MC4	ADC4 MC3	ADC4 MC2	ADC4 MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC4TERST**: ADC trigger 4 on timer E reset and counter roll-over
Refer to ADC4TCRST description.

Bit 30 **ADC4TEC4**: ADC trigger 4 on timer E compare 4
Refer to ADC4TAC2 description.

Bit 29 **ADC4TEC3**: ADC trigger 4 on timer E compare 3
Refer to ADC4TAC2 description.

Bit 28 **ADC4TEC2**: ADC trigger 4 on timer E compare 2
Refer to ADC4TAC2 description.

Bit 27 **ADC4TDRST**: ADC trigger 4 on timer D reset and counter roll-over
Refer to ADC4TCRST description.

Bit 26 **ADC4TDPER**: ADC trigger 4 on timer D period
Refer to ADC4TAPER description.

Bit 25 **ADC4TDC4**: ADC trigger 4 on timer D compare 4
Refer to ADC4TAC2 description.

Bit 24 **ADC4TFPER**: ADC trigger 4 on timer F period
Refer to ADC4TAPER description.

Bit 23 **ADC4TDC2**: ADC trigger 4 on timer D compare 2
Refer to ADC4TAC2 description.

Bit 22 **ADC4TCRST**: ADC trigger 4 on timer C reset and counter roll-over
This bit enables the generation of an ADC trigger upon timer C reset and roll-over event, on ADC trigger 1 output.

Bit 21 **ADC4TCPER**: ADC trigger 4 on timer C period
Refer to ADC4TAPER description.

Bit 20 **ADC4TCC4**: ADC trigger 4 on timer C compare 4
Refer to ADC4TAC2 description.

Bit 19 **ADC4TFC4**: ADC trigger 4 on timer F compare 4
Refer to ADC4TAC2 description.

Bit 18 **ADC4TCC2**: ADC trigger 4 on timer C compare 2
Refer to ADC4TAC2 description.

Bit 17 **ADC4TBPER**: ADC trigger 4 on timer B period
Refer to ADC4TAPER description.

Bit 16 **ADC4TBC4**: ADC trigger 4 on timer B compare 4
Refer to ADC4TAC2 description.

Bit 15 **ADC4TFC3**: ADC trigger 4 on timer F compare 3
Refer to ADC4TAC2 description.

Bit 14 **ADC4TBC2**: ADC trigger 4 on timer B compare 2
Refer to ADC4TAC2 description.

Bit 13 **ADC4TAPER**: ADC trigger 4 on timer A period
This bit enables the generation of an ADC trigger upon timer A event, on ADC trigger 2 output.

Bit 12 **ADC4TAC4**: ADC trigger 4 on timer A compare 4
Refer to ADC4TAC2 description.

Bit 11 **ADC4TFC2**: ADC trigger 4 on timer F compare 2
Refer to ADC4TAC2 description.

Bit 10 **ADC4TAC2**: ADC trigger 4 on timer A compare 2
This bit enables the generation of an ADC trigger upon timer A compare 2, on ADC trigger 2 output.

Bit 9 **ADC4EEV10**: ADC trigger 4 on external event 10
Refer to ADC4EEV6 description.

Bit 8 **ADC4EEV9**: ADC trigger 4 on external event 9
Refer to ADC4EEV6 description.

Bit 7 **ADC4EEV8**: ADC trigger 4 on external event 8
Refer to ADC4EEV6 description.

Bit 6 **ADC4EEV7**: ADC trigger 4 on external event 7
Refer to ADC4EEV6 description.

Bit 5 **ADC4EEV6**: ADC trigger 4 on external event 6
This bit enables the generation of an ADC trigger upon external event 6, on ADC trigger 2 output.

Bit 4 **ADC4MPER**: ADC trigger 4 on master period
This bit enables the generation of an ADC trigger upon master period event, on ADC trigger 2 output.

Bit 3 **ADC4MC4**: ADC trigger 4 on master compare 4
Refer to ADC4MC1 description.

Bit 2 **ADC4MC3**: ADC trigger 4 on master compare 3
Refer to ADC4MC1 description.

Bit 1 **ADC4MC2**: ADC trigger 4 on master compare 2
Refer to ADC4MC1 description.

Bit 0 **ADC4MC1**: ADC trigger 4 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 2 output.

34.5.71 HRTIM DLL control register (HRTIM_DLLCR)

Address offset: 0x3CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALRTE[1:0]		CALEN	CAL
												rw	rw	rw	wo

Bits 31:4 Reserved, must be kept at reset value

Bits 3:2 **CALRTE[1:0]**: DLL Calibration rate
This defines the DLL calibration periodicity.
00: 1048576 * t_{HRTIM} (3.495 ms for f_{HRTIM} = 300 MHz)
01: 131072 * t_{HRTIM} (437 µs for f_{HRTIM} = 300 MHz)
10: 16384 * t_{HRTIM} (55 µs for f_{HRTIM} = 300 MHz)
11: 2048 * t_{HRTIM} (6.8 µs for f_{HRTIM} = 300 MHz)

Bit 1 **CALEN**: DLL Calibration Enable
This bit enables the periodic DLL calibration, as per CALRTE[1:0] bit setting. When CALEN bit is reset, the calibration can be started in single-shot mode with CAL bit.
0: Periodic calibration disabled
1: Calibration is performed periodically, as per CALRTE[1:0] setting
Note: CALEN must not be set simultaneously with CAL bit

Bit 0 **CAL**: DLL Calibration Start
This bit starts the DLL calibration process. It is write-only.
0: No calibration request
1: Calibration start
Note: CAL must not be set simultaneously with CALEN bit

34.5.72 HRTIM fault input register 1 (HRTIM_FLTINR1)

Address offset: 0x3D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4LCK	FLT4F[3:0]				FLT4SRC[0]	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3SRC[0]	FLT3P	FLT3E
rwo	rw	rw	rw	rw	rw	rw	rw	rwo	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT2LCK	FLT2F[3:0]				FLT2SRC[0]	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC[0]	FLT1P	FLT1E
rwo	rw	rw	rw	rw	rw	rw	rw	rwo	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4LCK**: Fault 4 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 30:27 **FLT4F[3:0]**: Fault 4 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 26 FLT4SRC[0]: Fault 4 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 25 **FLT4P**: Fault 4 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 24 **FLT4E**: Fault 4 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 23 **FLT3LCK**: Fault 3 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 22:19 **FLT3F[3:0]**: Fault 3 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 18 FLT3SRC[0]: Fault 3 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 17 **FLT3P**: Fault 3 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 16 **FLT3E**: Fault 3 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 15 **FLT2LCK**: Fault 2 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 14:11 **FLT2F[3:0]**: Fault 2 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 10 FLT2SRC[0]: Fault 2 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 9 **FLT2P**: Fault 2 polarity
Refer to FLT2P description in HRTIM_FLTINR2 register.

Bit 8 **FLT2E**: Fault 2 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 7 **FLT1LCK**: Fault 1 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 6:3 **FLT1F[3:0]**: Fault 1 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 2 **FLT1SRC[0]**: Fault 1 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 1 **FLT1P**: Fault 1 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 0 **FLT1E**: Fault 1 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

34.5.73 HRTIM fault input register 2 (HRTIM_FLTINR2)

Address offset: 0x3D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		Res.	Res.	FLT6 SRC[1]	FLT5 SRC[1]	FLT4 SRC[1]	FLT3 SRC[1]	FLT2 SRC[1]	FLT1 SRC[1]
						rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT6 LCK	FLT6F[3:0]				FLT6 SRC[0]	FLT6P	FLT6E	FLT5 LCK	FLT5F[3:0]				FLT5 SRC[0]	FLT5P	FLT5E
rwo	rw	rw	rw	rw	rw	rw	rw	rwo	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **FLTSD[1:0]**: Fault sampling clock division
This bit field indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the fault signal sampling clock (f_{FLTS}) used by the digital filters.
00: $f_{FLTS}=f_{HRTIM}$
01: $f_{FLTS}=f_{HRTIM}$ / 2
10: $f_{FLTS}=f_{HRTIM}$ / 4
11: $f_{FLTS}=f_{HRTIM}$ / 8
Note: This bit field must be written prior to any of the FLTxE enable bits.

Bits 23:21 Reserved, must be kept at reset value.

Bit 21 **FLT6SRC[1]**: Fault 6 source bit 1
Refer to FLT5SRC[0] description.

Bit 20 **FLT5SRC[1]**: Fault 5 source bit 1
Refer to FLT5SRC[0] description.

Bit 19 **FLT4SRC[1]**: Fault 4 source bit 1
Refer to FLT5SRC[0] description.

Bit 18 **FLT3SRC[1]**: Fault 3 source bit 1
Refer to FLT5SRC[0] description.

Bit 17 **FLT2SRC[1]**: Fault 2 source bit 1
Refer to FLT5SRC[0] description.

Bit 16 **FLT1SRC[1]**: Fault 1 source bit 1
Refer to FLT5SRC[0] description.

Bit 15 **FLT6LCK**: Fault 6 lock
Refer to FLT5LCK description.

Bits 14:11 **FLT6F[3:0]**: Fault 6 filter
Refer to FLT5F[3:0] description.

Bit 10 **FLT6SRC[0]**: Fault 6 source bit 0
Refer to FLT5SRC[0] description.

Bit 9 **FLT6P**: Fault 6 polarity
Refer to FLT5P description.

Bit 8 **FLT6E**: Fault 6 enable
Refer to FLT5E description.

Bit 7 **FLT5LCK**: Fault 5 lock
The FLT5LCK bit modifies the write attributes of the fault programming bit, so that they are protected against spurious write accesses.
This bit is write-once. Once it has been set, it cannot be modified till the next system or IP reset.
0: FLT5E, FLT5P, FLT5SRC[1:0], FLT5F[3:0] and FLT5BLK bits are read/write.
1: FLT5E, FLT5P, FLT5SRC[1:0], FLT5F[3:0] and FLT5BLK bits cannot be written (read-only mode)

Bits 6:3 **FLT5F[3:0]**: Fault 5 filter
This bit field defines the frequency used to sample FLT5 input and the length of the digital filter applied to FLT5. The digital filter is made of an event counter in which N events are needed to validate a transition on the output.
0000: No filter, FLT5 acts asynchronously
0001: $f_{SAMPLING} = f_{HRTIM}$, N = 2
0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4
0011: $f_{SAMPLING} = f_{HRTIM}$, N = 8
0100: $f_{SAMPLING} = f_{FLTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{FLTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{FLTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{FLTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{FLTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{FLTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{FLTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{FLTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{FLTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{FLTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{FLTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{FLTS}/32$, N = 8

Note: This bit field is written only when FLT5E enable bit is reset.
This bit field is modified when FLT5LOCK has been programmed.

Bit 2 **FLT5SRC[0]**: Fault 5 source bit 0
The FTL5SRC[1:0] bit field selects the FAULT5 input source (refer to the HRTIM fault input tables of *Chapter 5: Device configuration* for connection details).
00: Fault 5 input is HRTIM_FLT5 input pin
01: Fault 5 input is connected to a COMPx output
10: Fault 5 input is EEV5_muxout input pin
11: Fault 5 input is HRTIMx_EEV5_muxout input pin
Note: This bit field is written only when FLT5E enable bit is reset.

Bit 1 **FLT5P**: Fault 5 polarity
This bit selects the FAULT5 input polarity.
0: Fault 5 input is active low
1: Fault 5 input is active high
Note: This bit field is written only when FLT5E enable bit is reset.

Bit 0 **FLT5E**: Fault 5 enable
This bit enables the global FAULT5 input circuitry.
0: Fault 5 input disabled
1: Fault 5 input enabled

34.5.74 HRTIM burst DMA master timer update register (HRTIM_BDMUPR)

Address offset: 0x3D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MCMP4**: MCMP4R register update enable
Refer to MCR description.

Bit 8 **MCMP3**: MCMP3R register update enable
Refer to MCR description.

Bit 7 **MCMP2**: MCMP2R register update enable
Refer to MCR description.

Bit 6 **MCMP1**: MCMP1R register update enable
Refer to MCR description.

Bit 5 **MREP**: MREP register update enable
Refer to MCR description.

Bit 4 **MPER**: MPER register update enable
Refer to MCR description.

Bit 3 **MCNT**: MCNTR register update enable
Refer to MCR description.

Bit 2 **MDIER**: MDIER register update enable
Refer to MCR description.

Bit 1 **MICR**: MICR register update enable
Refer to MCR description.

Bit 0 **MCR**: MCR register update enable
This bit defines if the master timer MCR register is part of the list of registers to be updated by the burst DMA.
0: MCR register is not updated by burst DMA accesses
1: MCR register is updated by burst DMA accesses

34.5.75 HRTIM burst DMA timer x update register (HRTIM_BDTxUPR) (x = A to F)

Address offset: Block A: 0x3DC

Address offset: Block B: 0x3E0

Address offset: Block C: 0x3E4

Address offset: Block D: 0x3E8

Address offset: Block E: 0x3EC

Address offset: Block F: 0x3F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMx EEFR3	TIMx CR2	TIMxFL TR	TIMxO UTR	TIMxC HPR	TIMxR STR	TIMxE EFR2
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMxE EFR1	TIMxR ST2R	TIMxS ET2R	TIMxR ST1R	TIMxS ET1R	TIMxD TxR	TIMxC MP4	TIMxC MP3	TIMxC MP2	TIMxC MP1	TIMxR EP	TIMxP ER	TIMxC NT	TIMxDI ER	TIMxIC R	TIMxC R
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **TIMxEEFR3**: HRTIM_EEFxR3 register update enable
Refer to TIMxCR description.

Bit 21 **TIMxCR2**: HRTIM_TIMxCR2 register update enable
Refer to TIMxCR description.

Bit 20 **TIMxFLTR**: HRTIM_FLTxR register update enable
Refer to TIMxCR description.

Bit 19 **TIMxOUTR**: HRTIM_OUTxR register update enable
Refer to TIMxCR description.

Bit 18 **TIMxCHPR**: HRTIM_CHPxR register update enable
Refer to TIMxCR description.

Bit 17 **TIMxRSTR**: HRTIM_RSTxR register update enable
Refer to TIMxCR description.

Bit 16 **TIMxEEFR2**: HRTIM_EEFxR2 register update enable
Refer to TIMxCR description.

Bit 15 **TIMxEEFR1**: HRTIM_EEFxR1 register update enable
Refer to TIMxCR description.

Bit 14 **TIMxRST2R**: HRTIM_RST2xR register update enable
Refer to TIMxCR description.

Bit 13 **TIMxSET2R**: HRTIM_SET2xR register update enable
Refer to TIMxCR description.

Bit 12 **TIMxRST1R**: HRTIM_RST1xR register update enable
Refer to TIMxCR description.

Bit 11 **TIMxSET1R**: HRTIM_SET1xR register update enable
Refer to TIMxCR description.

Bit 10 **TIMxDTR**: HRTIM_DTxR register update enable
Refer to TIMxCR description.

Bit 9 **TIMxCMP4**: HRTIM_CMP4xR register update enable
Refer to TIMxCR description.

Bit 8 **TIMxCMP3**: HRTIM_CMP3xR register update enable
Refer to TIMxCR description.

Bit 7 **TIMxCMP2**: HRTIM_CMP2xR register update enable
Refer to TIMxCR description.

Bit 6 **TIMxCMP1**: HRTIM_CMP1xR register update enable
Refer to TIMxCR description.

Bit 5 **TIMxREP**: HRTIM_REPxR register update enable
Refer to TIMxCR description.

Bit 4 **TIMxPER**: HRTIM_PERxR register update enable
Refer to TIMxCR description.

Bit 3 **TIMxCNT**: HRTIM_CNTxR register update enable
Refer to TIMxCR description.

Bit 2 **TIMxDIER**: HRTIM_TIMxDIER register update enable
Refer to TIMxCR description.

Bit 1 **TIMxICR**: HRTIM_TIMxICR register update enable
Refer to TIMxCR description.

Bit 0 **TIMxCR**: HRTIM_TIMxCR register update enable
This bit defines if the master timer MCR register is part of the list of registers to be updated by the burst DMA.
0: HRTIM_TIMxCR register is not updated by burst DMA accesses
1: HRTIM_TIMxCR register is updated by burst DMA accesses

34.5.76 HRTIM burst DMA data register (HRTIM_BDMADR)

Address offset: 0x3F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BDMADR[31:16]															
wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BDMADR[15:0]															
wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo

Bits 31:0 **BDMADR[31:0]**: Burst DMA data register

Write accesses to this register triggers:

– the copy of the data value into the registers enabled in BDTxUPR and BDMUPR register bits

– the increment of the register pointer to the next location to be filled

34.5.77 HRTIM ADC extended trigger register (HRTIM_ADCER)

Address offset: 0x3F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ADC10TRG[4:0]					ADC9TRG[4:0]					ADC8TRG[4:0]				
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADC7TRG[4:0]					ADC6TRG[4:0]					ADC5TRG[4:0]				
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **ADC10TRG[4:0]**: ADC trigger 10 selection

This bit selects the ADC trigger 10 source.
Refer to ADC6TRG[4:0] description.

Bits 25:21 **ADC9TRG[4:0]**: ADC trigger 9 selection

This bit selects the ADC trigger 9 source.
Refer to ADC5TRG[4:0] description.

Bits 20:16 **ADC8TRG[4:0]**: ADC trigger 8 selection

This bit selects the ADC trigger 8 source.
Refer to ADC6TRG[4:0] description.

Bit 15 Reserved, must be kept at reset value.

Bits 14:10 **ADC7TRG[4:0]**: ADC trigger 7 selection

This bit selects the ADC trigger 7 source.
Refer to ADC5TRG[4:0] description.

Bits 9:5 **ADC6TRG[4:0]**: ADC trigger 6 selection

This bit selects the ADC trigger 6 source.

0: Trigger on master compare 1
1: Trigger on master compare 2
2: Trigger on master compare 3
3: Trigger on master compare 4
4: Trigger on master period
5: Trigger on external event 6
6: Trigger on external event 7
7: Trigger on external event 8
8: Trigger on external event 9
9: Trigger on external event 10
10: Trigger on timer A compare 2
11: Trigger on timer A compare 4
12: Trigger on timer A period
13: Trigger on timer B compare 2
14: Trigger on timer B compare 4
15: Trigger on timer B period
16: Trigger on timer C compare 2
17: Trigger on timer C compare 4
18: Trigger on timer C period
19: Trigger on timer C reset and counter roll-over
20: Trigger on timer D compare 2
21: Trigger on timer D compare 4
22: Trigger on timer D period
23: Trigger on timer D reset and counter roll-over
24: Trigger on timer E compare 2
25: Trigger on timer E compare 3
26: Trigger on timer E compare 4
27: Trigger on timer E reset and counter roll-over
28: Trigger on timer F compare 2
29: Trigger on timer F compare 3
30: Trigger on timer F compare 4
31: Trigger on timer F period

Bits 4:0 **ADC5TRG[4:0]**: ADC trigger 5 selection
This bit selects the ADC trigger 5 source.
0: Trigger on master compare 1
1: Trigger on master compare 2
2: Trigger on master compare 3
3: Trigger on master compare 4
4: Trigger on master period
5: Trigger on external event 1
6: Trigger on external event 2
7: Trigger on external event 3
8: Trigger on external event 4
9: Trigger on external event 5
10: Trigger on timer A compare 3
11: Trigger on timer A compare 4
12: Trigger on timer A period
13: Trigger on timer A reset and counter roll-over
14: Trigger on timer B compare 3
15: Trigger on timer B compare 4
16: Trigger on timer B period
17: Trigger on timer B reset and counter roll-over
18: Trigger on timer C compare 3
19: Trigger on timer C compare 4
20: Trigger on timer C period
21: Trigger on timer D compare 3
22: Trigger on timer D compare 4
23: Trigger on timer D period
24: Trigger on timer E compare 3
25: Trigger on timer E compare 4
26: Trigger on timer E period
27: Trigger on timer F compare 2
28: Trigger on timer F compare 3
29: Trigger on timer F compare 4
30: Trigger on timer F period
31: Trigger on timer F reset and counter roll-over

34.5.78 HRTIM ADC trigger update register (HRTIM_ADCUR)

Address offset: 0x3FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AD10USRC[2:0]			Res.	AD9USRC[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AD8USRC[2:0]			Res.	AD7USRC[2:0]			Res.	AD6USRC[2:0]			Res.	AD5USRC[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **AD10USRC[2:0]**: ADC trigger 10 update source
Refer to AD5USRC[2:0] description.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **AD9USRC[2:0]**: ADC trigger 9 update source
Refer to AD5USRC[2:0] description.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **AD8USRC[2:0]**: ADC trigger 8 update source
Refer to AD5USRC[2:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **AD7USRC[2:0]**: ADC trigger 7 update source
Refer to AD5USRC[2:0] description.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **AD6USRC[2:0]**: ADC trigger 6 update source
Refer to AD5USRC[2:0] description.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **AD5USRC[2:0]**: ADC trigger 5 update source
These bits define the source which triggers the update of the ADC5TRG[4:0] bit field in the HRTIM_ADCER register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, with the BRSTDMA[1:0] bit field in HRTIM_MCR or the UPDGAT[3:0] bit field in HRTIM_TIMxCR register.
000: Master timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110: Timer F
111: Reserved

34.5.79 HRTIM ADC post scaler register 1 (HRTIM_ADCPS1)

Address offset: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC5PSC[4:0]					Res.	ADC4PSC[4:0]					Res.	ADC3PSC[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3PSC[3:0]				Res.	ADC2PSC[4:0]					Res.	ADC1PSC[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADC5PSC[4:0]**: ADC 5 post scaler
Refer to ADC1PSC[4:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **ADC4PSC[4:0]**: ADC 4 post scaler
Refer to ADC1PSC[4:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **ADC3PSC[4:0]**: ADC 3 post scaler
Refer to ADC1PSC[4:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **ADC2PSC[4:0]**: ADC 2 post scaler
Refer to ADC1PSC[4:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **ADC1PSC[4:0]**: ADC 1 post scaler
This bit selects the ADC 1 Post scaler ratio.

34.5.80 HRTIM ADC post scaler register 2 (HRTIM_ADCPS2)

Address offset: 0x404

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC10PSC[4:0]					Res.	ADC9PSC[4:0]					Res.	ADC8PSC[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC8PSC[3:0]				Res.	ADC7PSC[4:0]					Res.	ADC6PSC[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADC10PSC[4:0]**: ADC 10 post scaler
Refer to ADC1PSC[4:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **ADC9PSC[4:0]**: ADC 9 post scaler
Refer to ADC1PSC[4:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **ADC8PSC[4:0]**: ADC 8 post scaler
Refer to ADC1PSC[4:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **ADC7PSC[4:0]**: ADC 7 post scaler
Refer to ADC1PSC[4:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **ADC6PSC[4:0]**: ADC 6 post scaler
Refer to ADC1PSC[4:0] description.

34.5.81 HRTIM fault input register 3 (HRTIM_FLTINR3)

Address offset: 0x408

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4 RSTM	FLT4 CRES	FLT4CNT[3:0]				FLT4 BLKS	FLT4 BLKE	FLT3 RSTM	FLT3 CRES	FLT3CNT[3:0]				FLT3 BLKS	FLT3 BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT2 RSTM	FLT2 CRES	FLT2CNT[3:0]				FLT2 BLKS	FLT2 BLKE	FLT1 RSTM	FLT1 CRES	FLT1CNT[3:0]				FLT1 BLKS	FLT1 BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4RSTM**: Fault 4 reset mode
Refer to FLT5RSTM description.

Bit 30 **FLT4CRES**: Fault 4 counter reset
Refer to FLT5CRES description.

Bits 29:26 **FLT4CNT[3:0]**: Fault 4 counter
Refer to FLT5CNT description.

Bit 25 **FLT4BLKS**: Fault 4 blanking source
Refer to FLT5BLKS description.

Bit 24 **FLT4BLKE**: Fault 4 blanking enable
Refer to FLT5BLKE description.

Bit 23 **FLT3RSTM**: Fault 3 reset mode
Refer to FLT5RSTM description.

Bit 22 **FLT3CRES**: Fault 3 counter reset
Refer to FLT5CRES description.

Bits 21:18 **FLT3CNT[3:0]**: Fault 3 counter
Refer to FLT5CNT description.

Bit 17 **FLT3BLKS**: Fault 3 blanking source
Refer to FLT5BLKS description.

Bit 16 **FLT3BLKE**: Fault 3 blanking enable
Refer to FLT5BLKE description.

Bit 15 **FLT2RSTM**: Fault 2 reset mode
Refer to FLT5RSTM description.

Bit 14 **FLT2CRES**: Fault 2 counter reset
Refer to FLT5CRES description.

Bits 13:10 **FLT2CNT[3:0]**: Fault 2 counter
Refer to FLT5CNT description.

Bit 9 **FLT2BLKS**: Fault 2 blanking source
Refer to FLT5BLKS description.

Bit 8 **FLT2BLKE**: Fault 2 blanking enable
Refer to FLT5BLKE description.

Bit 7 **FLT1RSTM**: Fault 1 reset mode
Refer to FLT5RSTM description.

Bit 6 **FLT1CRES**: Fault 1 counter reset
Refer to FLT5CRES description.

Bits 5:2 **FLT1CNT[3:0]**: Fault 1 counter
Refer to FLT5CNT description.

Bit 1 **FLT1BLKS**: Fault 1 blanking source
Refer to FLT5BLKS description.

Bit 0 **FLT1BLKE**: Fault 1 blanking enable
Refer to FLT5BLKE description.

34.5.82 HRTIM fault input register 4 (HRTIM_FLTINR4)

Address offset: 0x40C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT6 RSTM	FLT6 CRES	FLT6CNT[3:0]				FLT6 BLKS	FLT6 BLKE	FLT5 RSTM	FLT5 CRES	FLT5CNT[3:0]				FLT5 BLKS	FLT5 BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLT6RSTM**: Fault 6 reset mode
Refer to FLT5RSTM description.

Bit 14 **FLT6CRES**: Fault 6 counter reset
Refer to FLT5CRES description.

Bits 13:10 **FLT6CNT[3:0]**: Fault 6 counter
Refer to FLT5CNT description.

Bit 9 **FLT6BLKS**: Fault 6 blanking source
Refer to FLT5BLKS description.

Bit 8 **FLT6BLKE**: Fault 6 blanking enable
Refer to FLT6BLKE description.

Bit 7 **FLT5RSTM**: Fault 5 reset mode
This bit selects the FAULT5 counter reset mode
0: Fault 5 counter is reset on each reset / roll-over event
1: Fault 5 counter is reset on each reset / roll-over event only if no fault occurred during last counting period.
This bit field is written only when FLT5E enable bit is reset.

Bit 6 **FLT5CRES**: Fault 5 counter reset
This bit resets the FAULT5 counter. It is set by software and reset by hardware.
0: No action
1: Fault 5 counter is reset

Bits 5:2 **FLT5CNT[3:0]**: Fault 5 counter

This bit field selects the FAULT5 counter threshold. A fault is considered valid when the number of events is equal to the (FLT5CNT[3:0]+1) value.

Bit 1 **FLT5BLKS**: Fault 5 blanking source

The FTL5BLKS bit selects the FAULT5 blanking source (refer to *Table 403* for details).

0: Fault 5 reset-aligned blanking window

1: Fault 5 Moving blanking window

Note: This bit field is written only when FLT5E enable bit is reset.

Bit 0 **FLT5BLKE**: Fault 5 blanking enable

The FTL5BLKE bit selects the FAULT5 blanking mode. The blanking source is defined by the FLT5BLKS bit.

0: No blanking on fault 5

1: Fault 5 blanking mode

Note: This bit field is written only when FLT5E enable bit is reset

35 Advanced-control timers (TIM1/TIM8)

35.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1/TIM8) and general-purpose (TIMx) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 35.3.30: Timer synchronization*.

35.2 TIM1/TIM8 main features

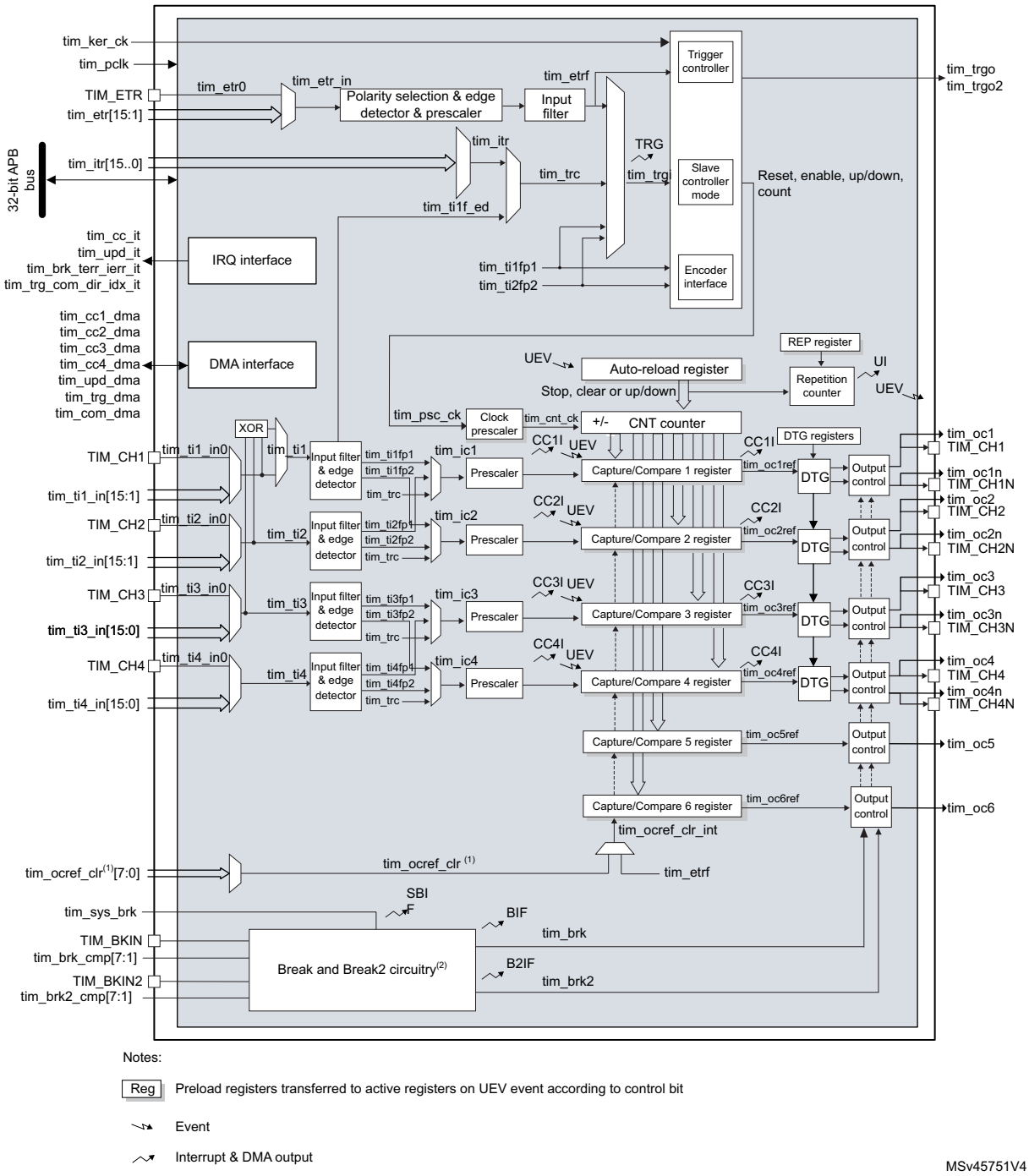
TIM1/TIM8 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input capture (but channels 5 and 6)
  - Output compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

35.3 TIM1/TIM8 functional description

35.3.1 Block diagram

Figure 424. Advanced-control timer block diagram



1. This feature is not available on all timers. Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals*.
2. Refer to *Figure 471: Break and Break2 circuitry overview* for details.

35.3.2 TIM1/TIM8 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 410. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input / Output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 411. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 411. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[7:1]	Input	Break input for internal signals
tim_brk2_cmp[7:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trg_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trg_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

The sources connected to the tim_ti[1..4] input multiplexers are listed in *Section 5.5.3.1: Interconnects to the tim_ti[1..4] input multiplexers* from Device configuration chapter.

The internal sources connected to the tim_itr input multiplexer and the tim_etr input multiplexer are listed in *Section 5.5.3.2: Interconnects to the trigger input multiplexers* from Device configuration chapter.

The sources connected to the tim_brk and tim_brk2inputs are listed in *Section 5.5.3.3: Interconnects to the break & break2 inputs* from Device configuration chapter.

The internal sources connected to the tim_ocref_clr input multiplexer are listed in *Section 5.5.3.4: Interconnects to the ocref_clr input multiplexer* from Device configuration chapter.

35.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 425 and *Figure 426* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 425. Counter timing diagram with prescaler division change from 1 to 2

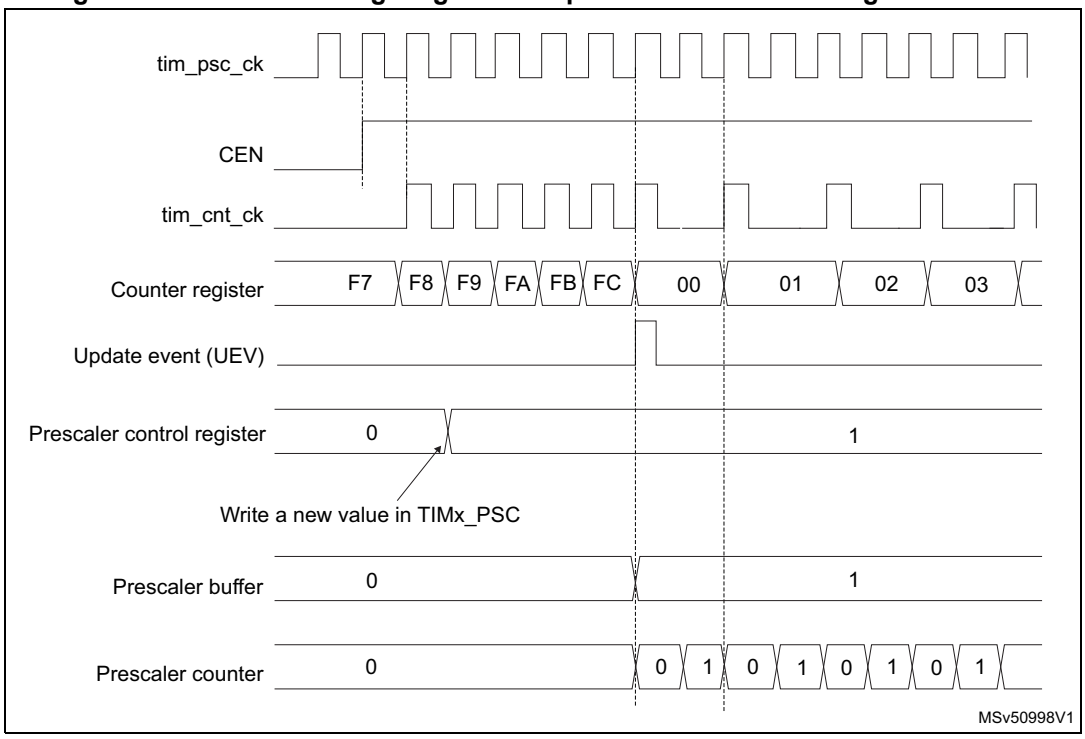
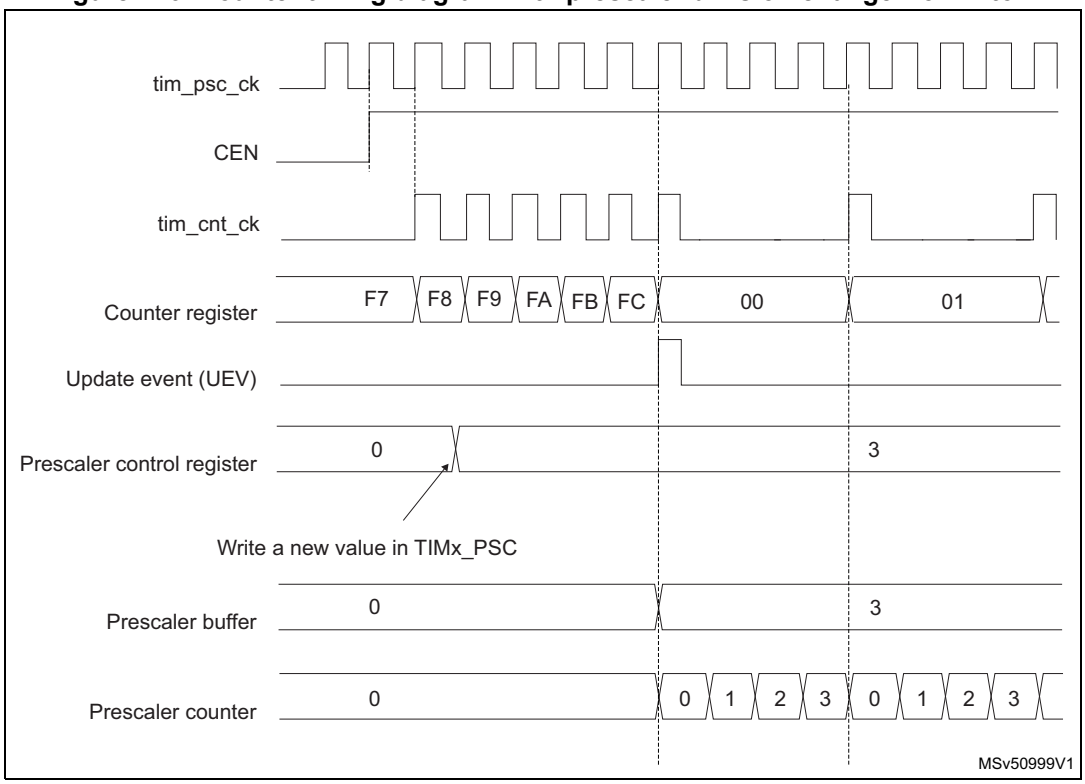


Figure 426. Counter timing diagram with prescaler division change from 1 to 4



35.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 427. Counter timing diagram, internal clock divided by 1

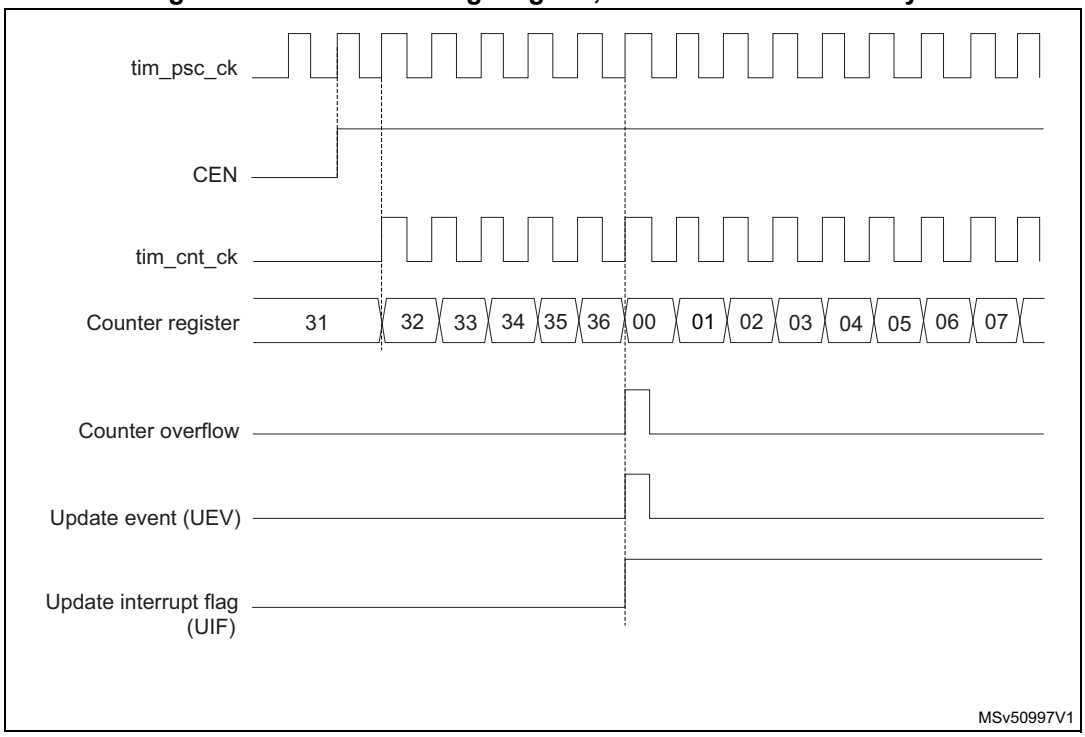


Figure 428. Counter timing diagram, internal clock divided by 2

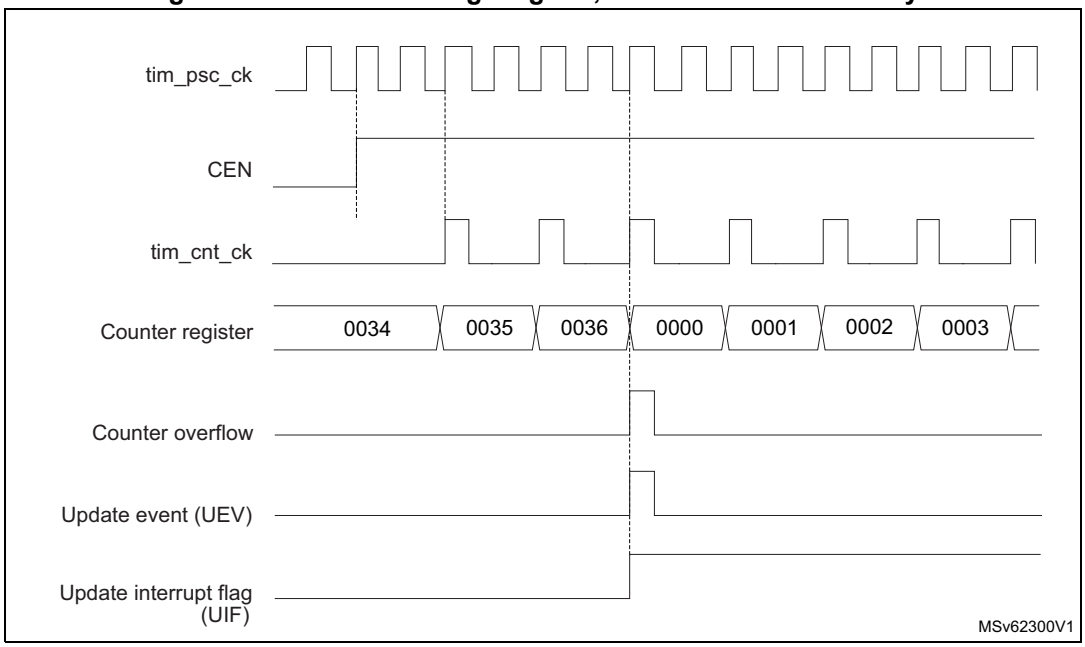


Figure 429. Counter timing diagram, internal clock divided by 4

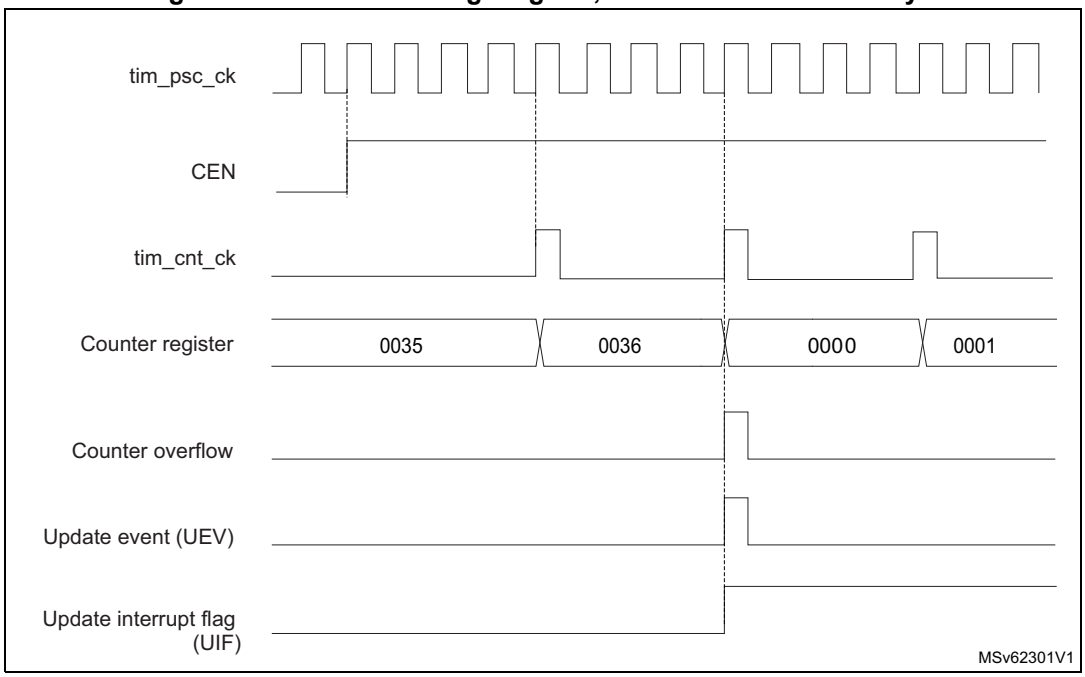


Figure 430. Counter timing diagram, internal clock divided by N

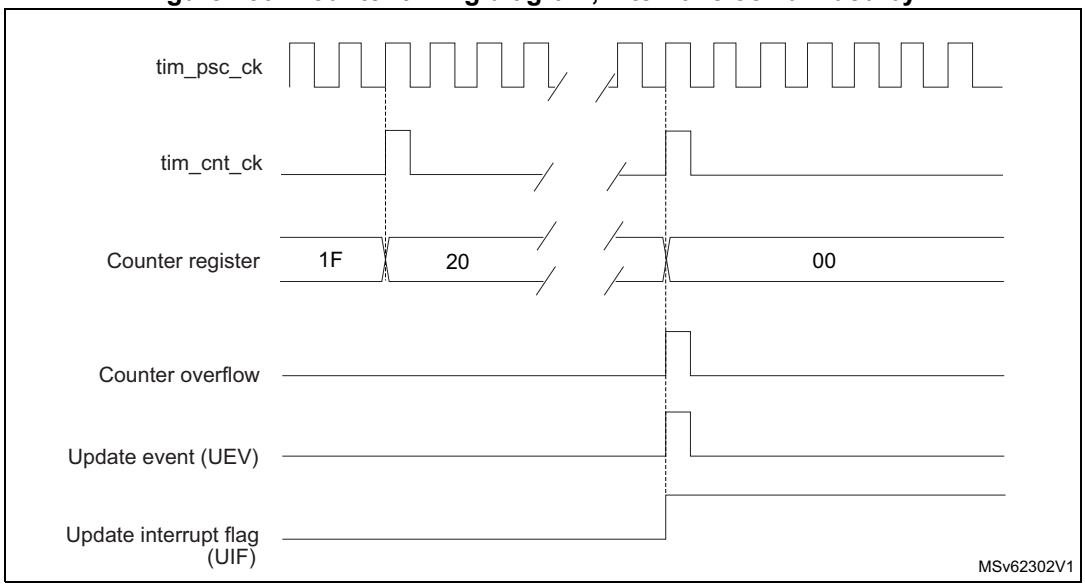


Figure 431. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

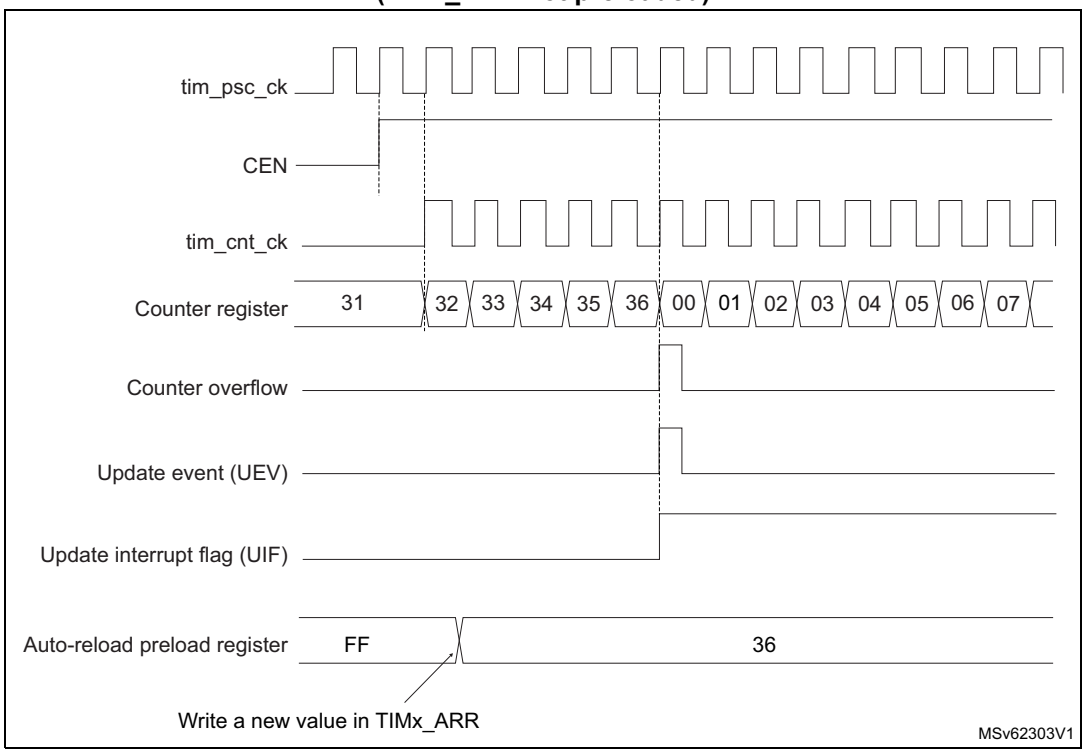
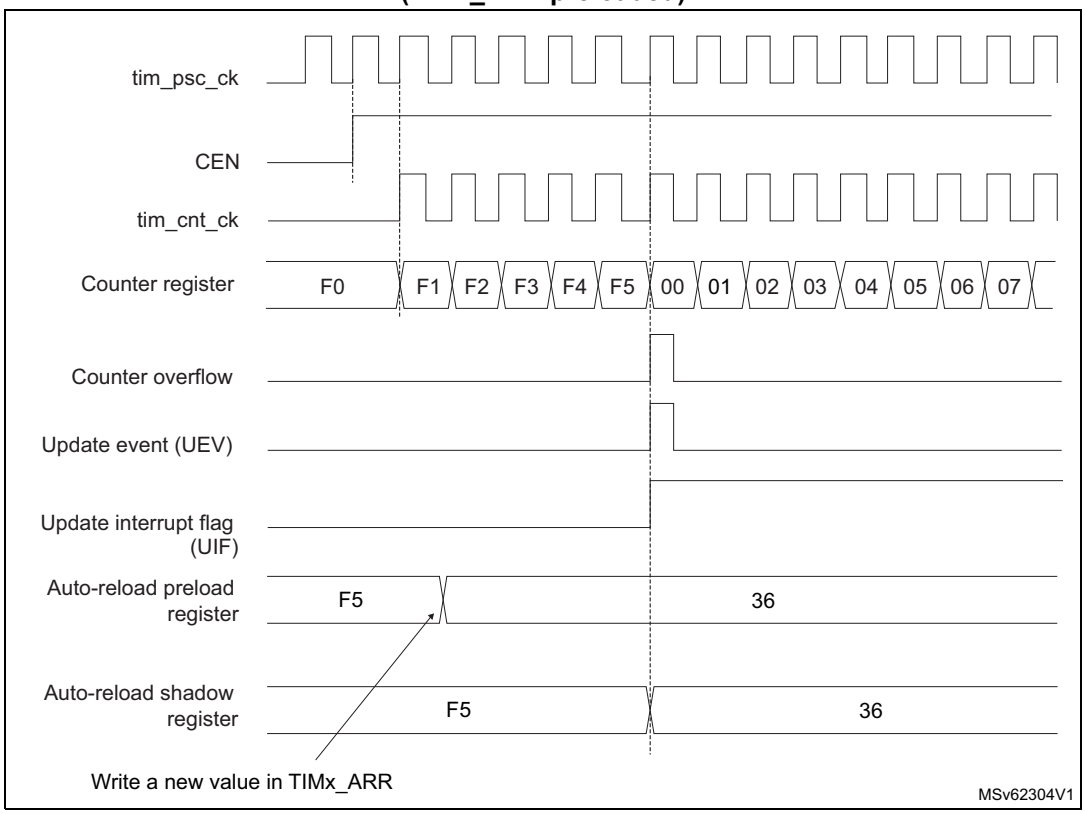


Figure 432. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 433. Counter timing diagram, internal clock divided by 1

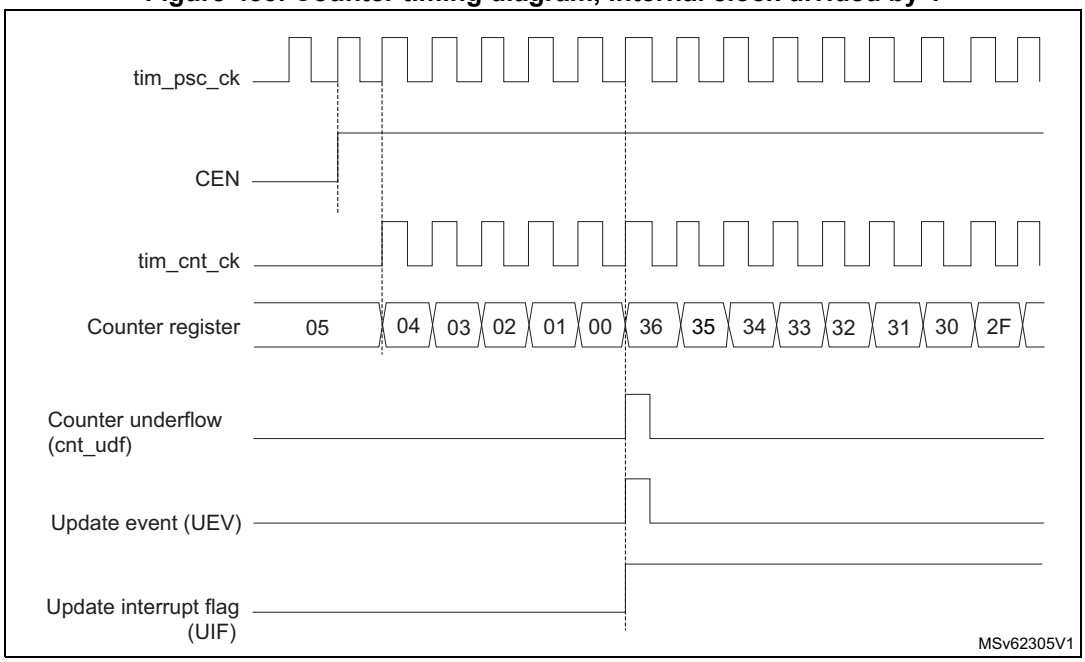


Figure 434. Counter timing diagram, internal clock divided by 2

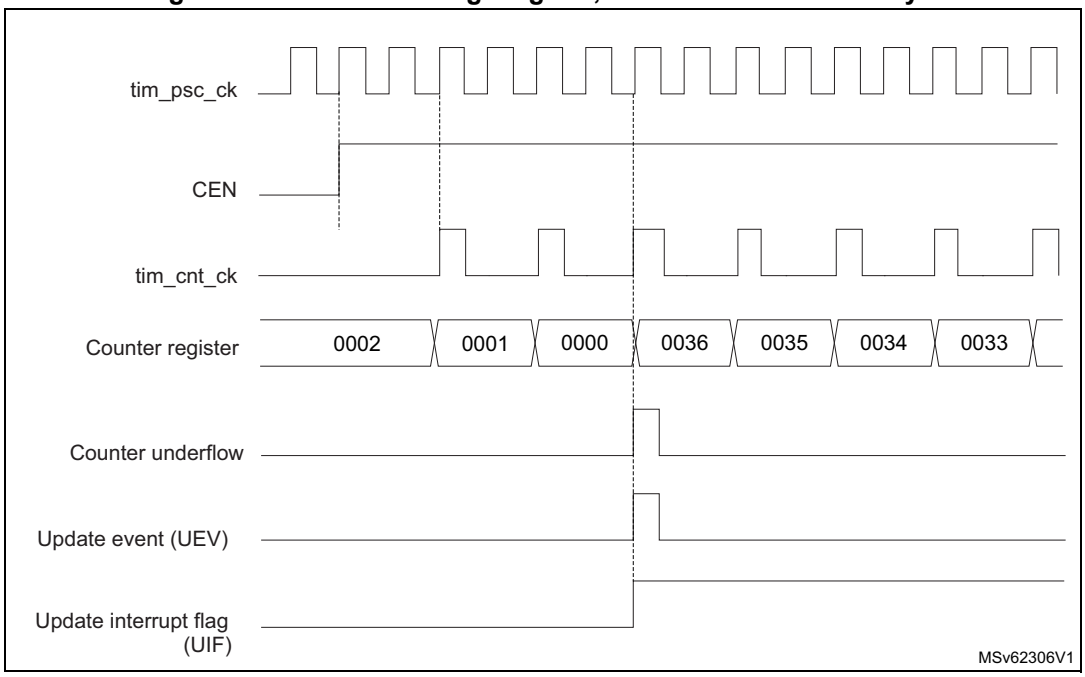


Figure 435. Counter timing diagram, internal clock divided by 4

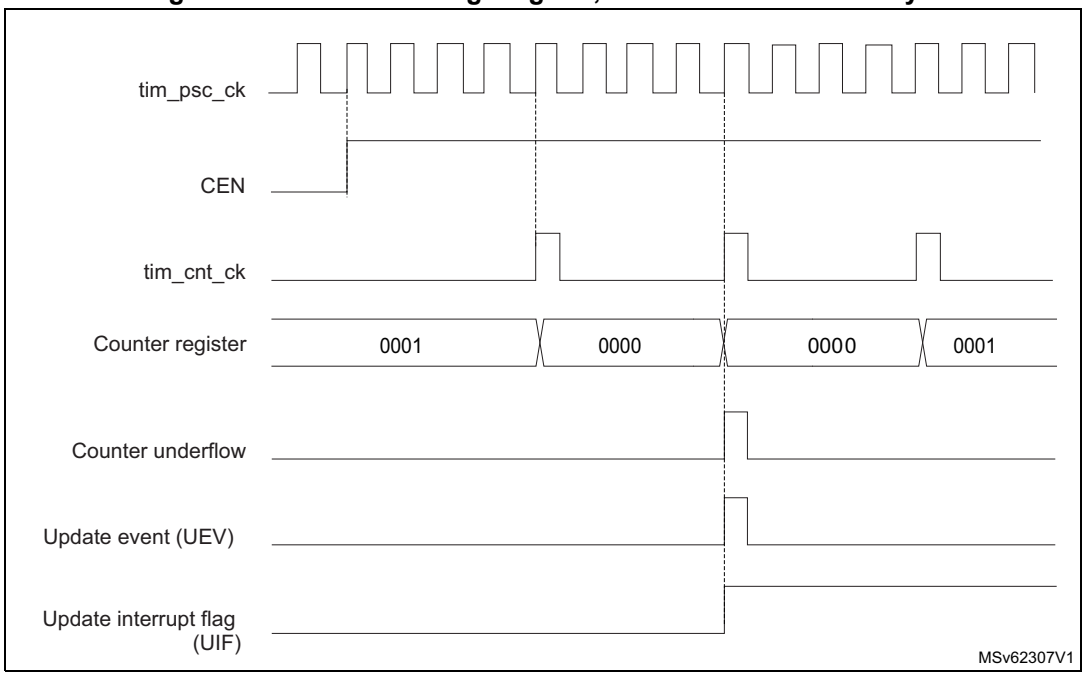


Figure 436. Counter timing diagram, internal clock divided by N

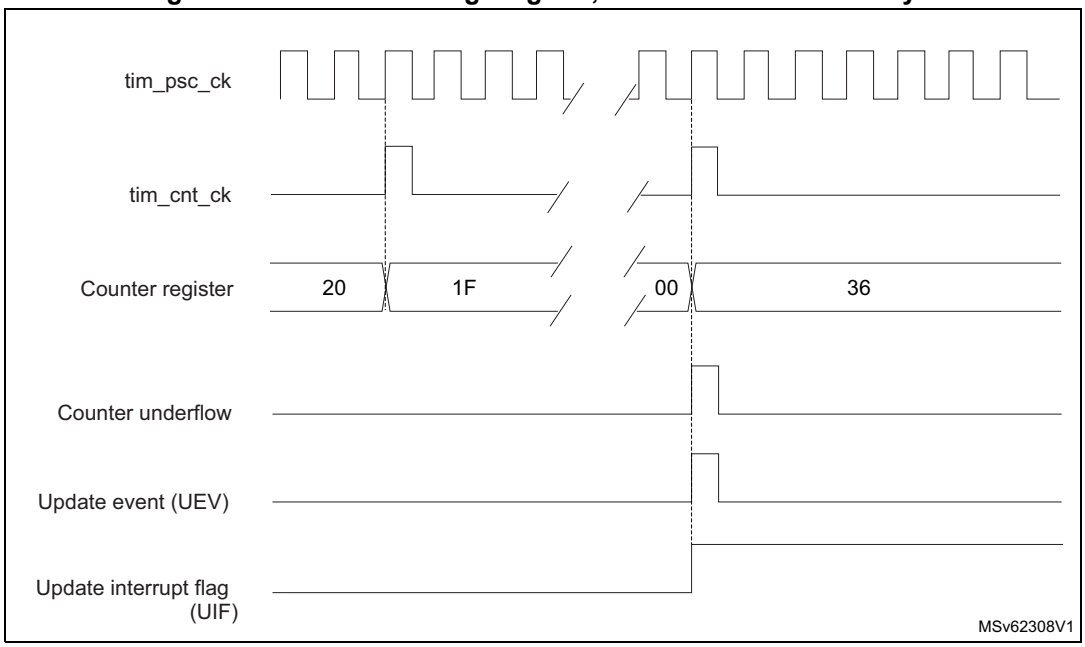
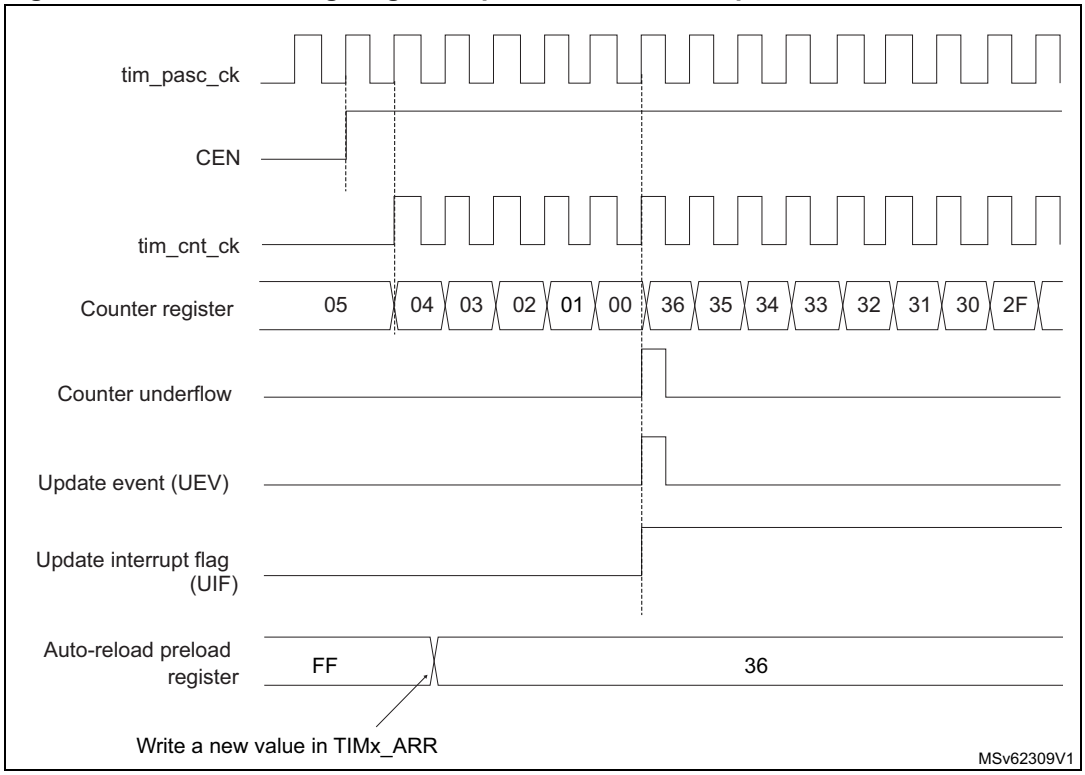


Figure 437. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-

reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller). In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

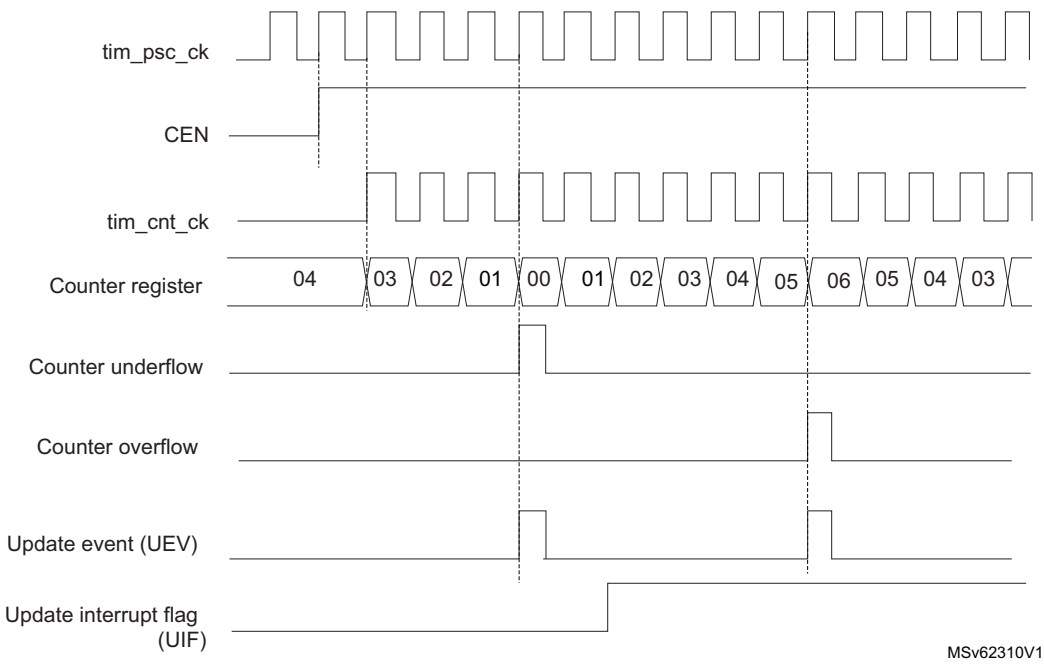
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 438. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



Note: Here, center-aligned mode 1 is used (for more details refer to *Section 35.6: TIM1/TIM8 registers*).

Figure 439. Counter timing diagram, internal clock divided by 2

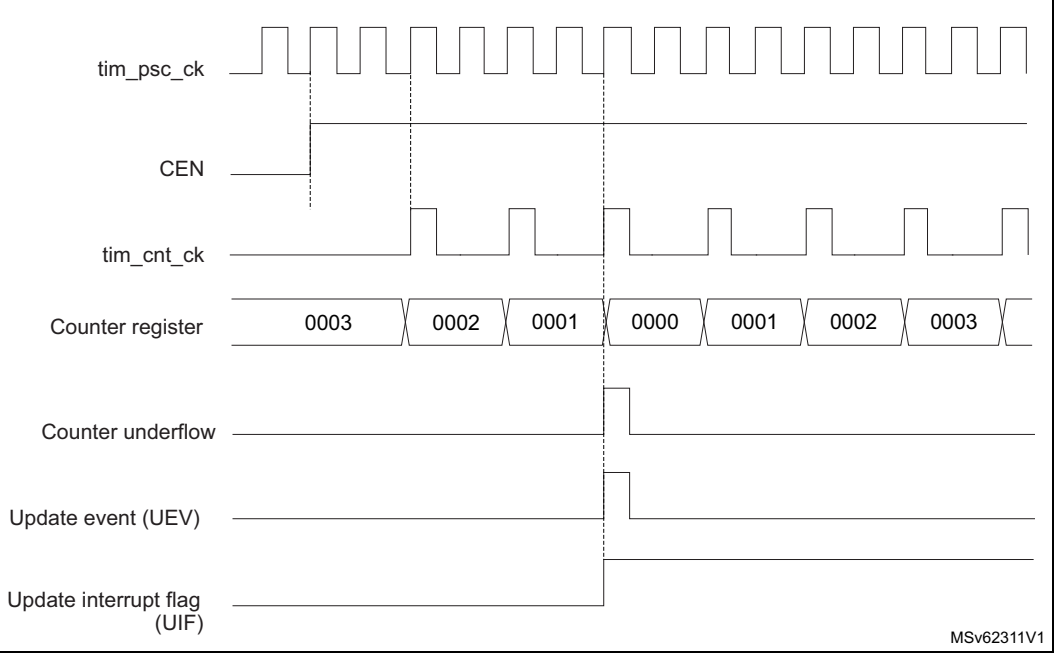


Figure 440. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

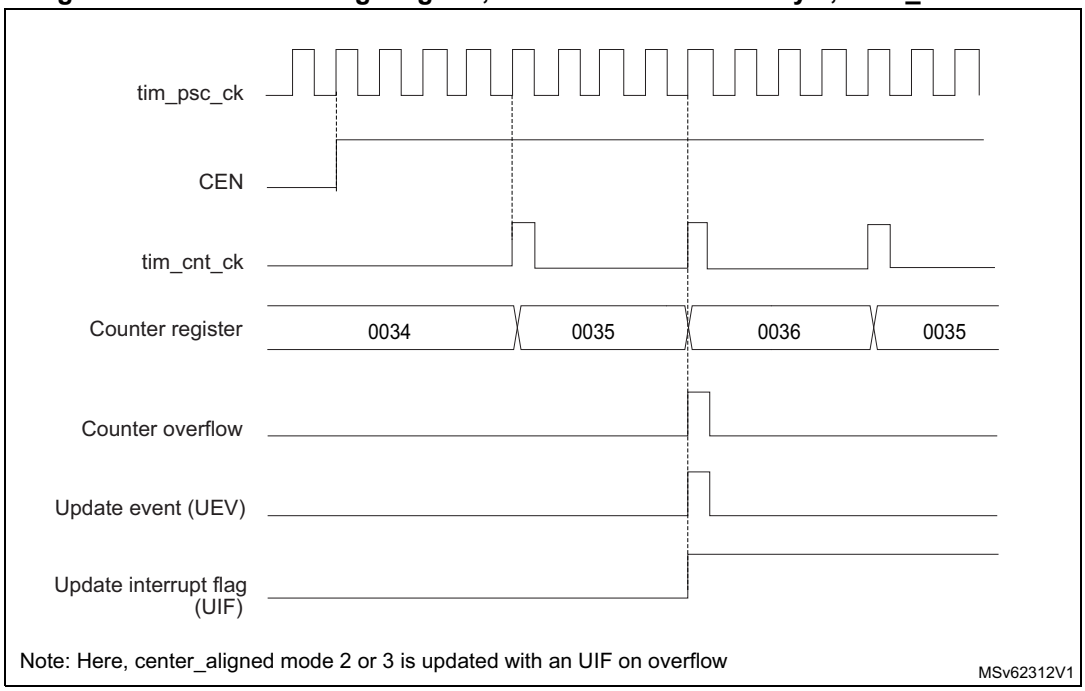


Figure 441. Counter timing diagram, internal clock divided by N

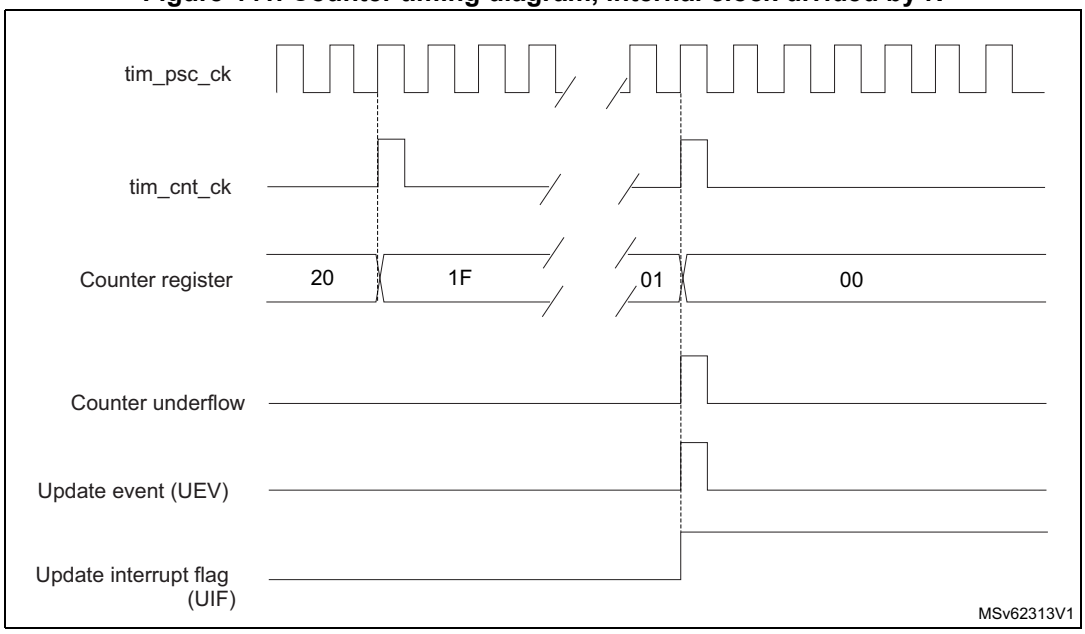


Figure 442. Counter timing diagram, update event with ARPE=1 (counter underflow)

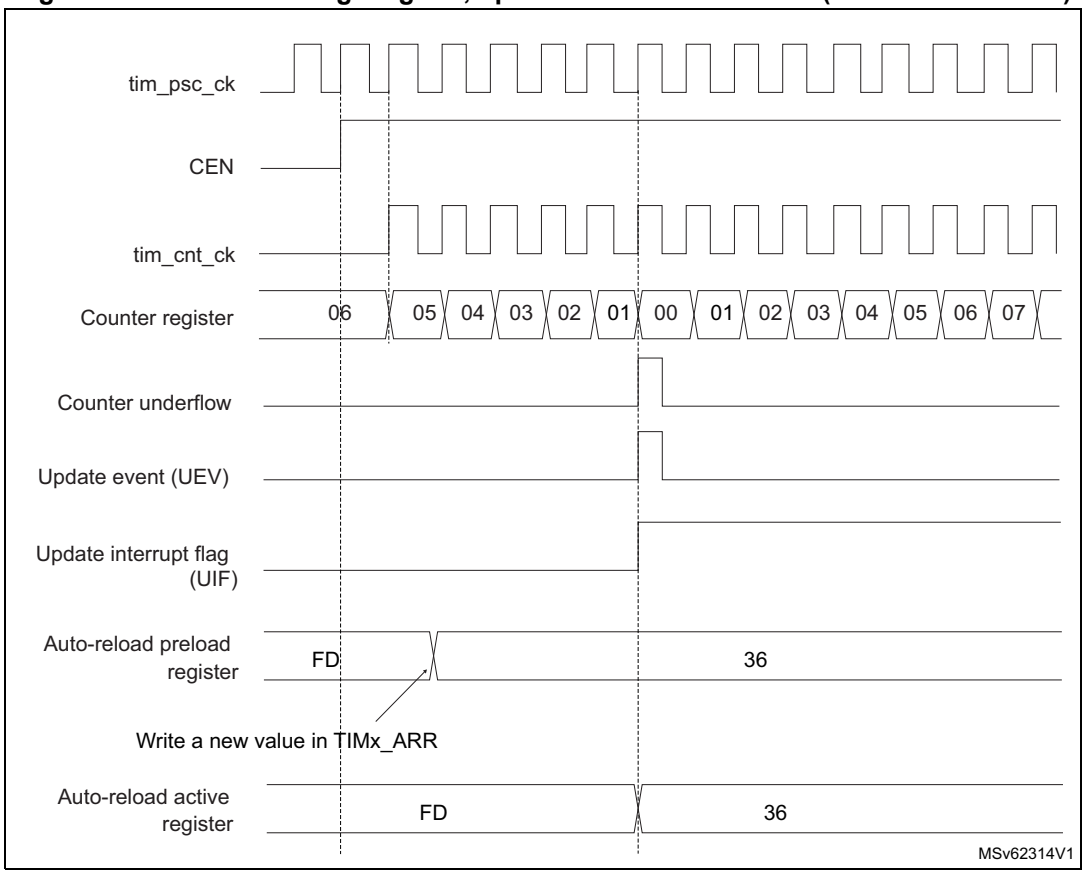
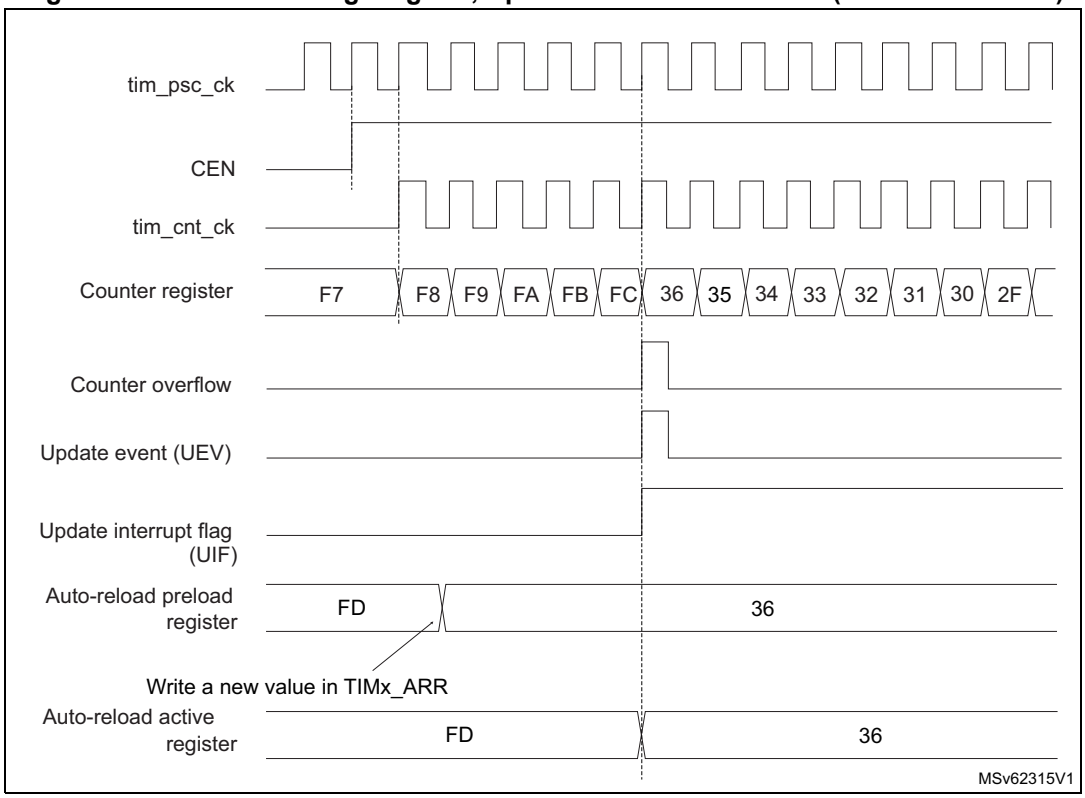


Figure 443. Counter timing diagram, Update event with ARPE=1 (counter overflow)



35.3.5 Repetition counter

Section 35.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

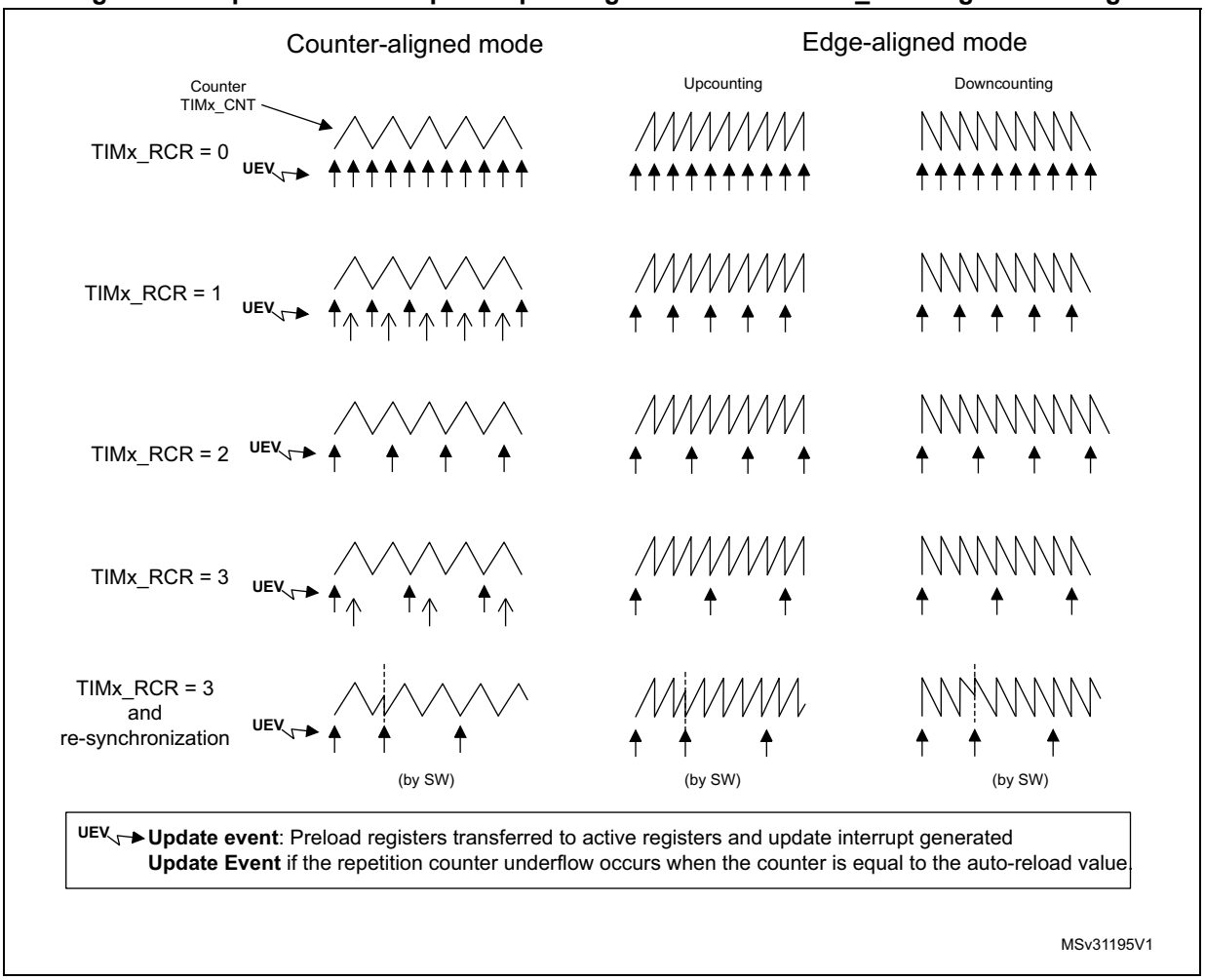
- At each counter overflow in upcounting mode
- At each counter underflow in downcounting mode
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 444*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 444. Update rate examples depending on mode and TIMx_RCR register settings



35.3.6 External trigger input

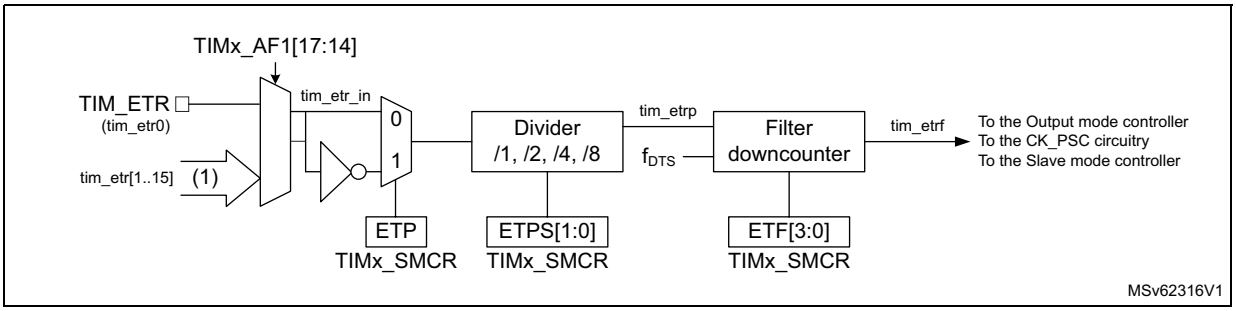
The timer features an external trigger input tim_etr_in. It can be used as:

- External clock (external clock mode 2, refer to *Section 35.3.7: Clock selection*)
- Trigger for the slave mode (refer to *Section 35.3.30: Timer synchronization*)
- PWM reset input for cycle-by-cycle current regulation (refer to *Section 35.3.9: Input capture mode*)

Figure 445 describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIMx_SMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bit field and digitally filtered with the ETF[3:0] bit field. The resulting signal

(tim_etrf) is available for three purposes: as an external clock, to condition the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 445. External trigger input block



The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bit field in the TIMx_AF1 register.

Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals* for the list of sources connected to the etr_in input in the product.

35.3.7 Clock selection

The counter clock can be provided by the following clock sources:

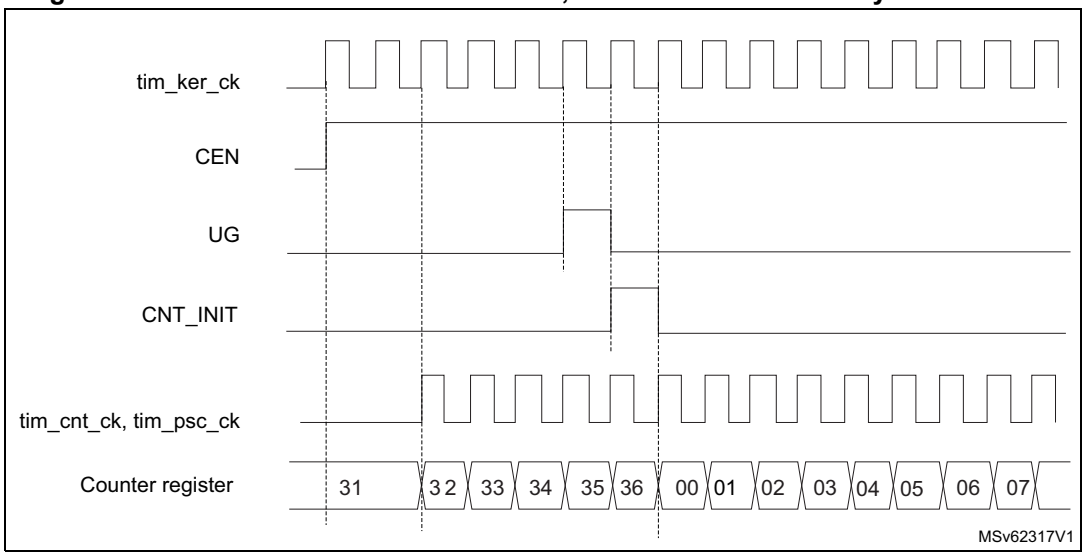
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 446 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

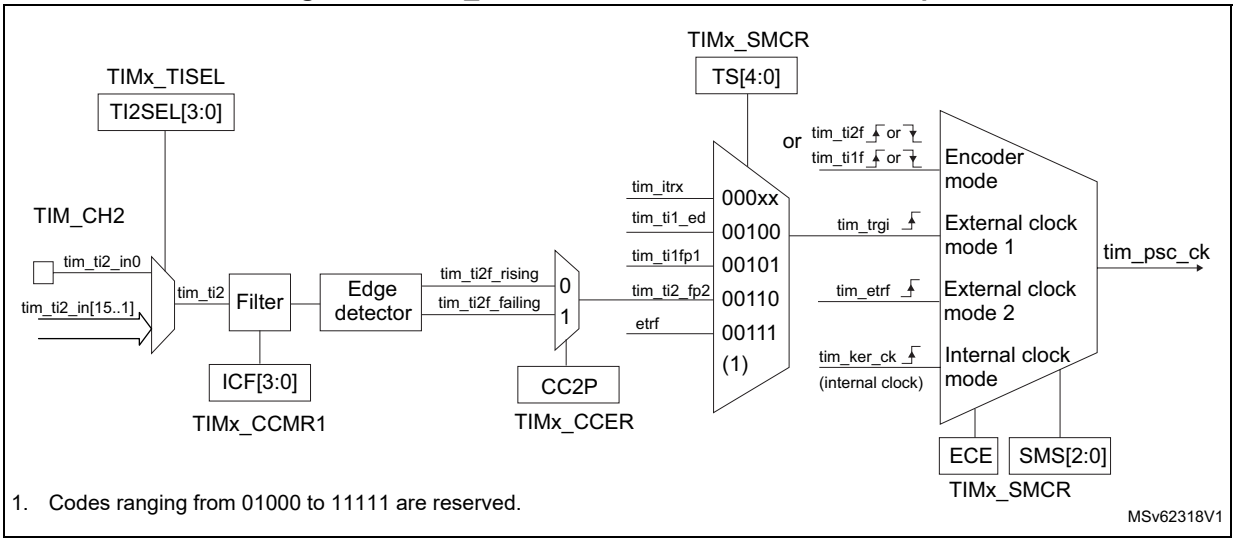
Figure 446. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 447. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

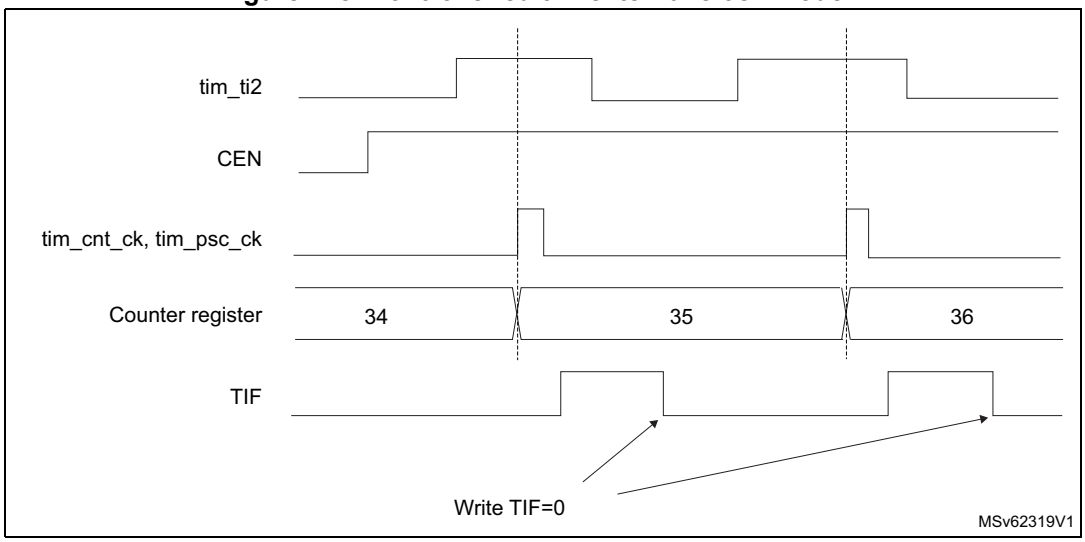
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 448. Control circuit in external clock mode 1



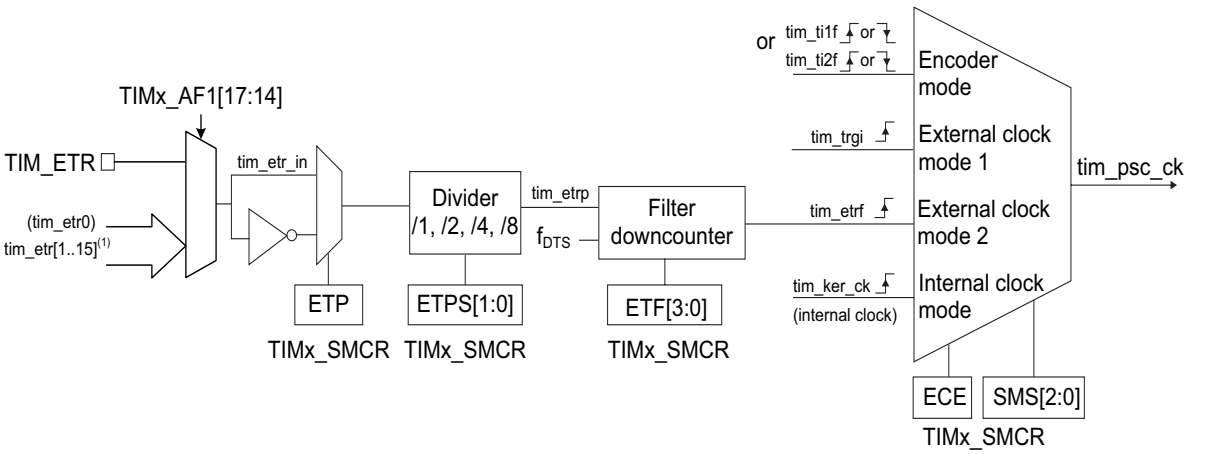
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

Figure 449 gives an overview of the external trigger input block.

Figure 449. External trigger input block



1. Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals*.

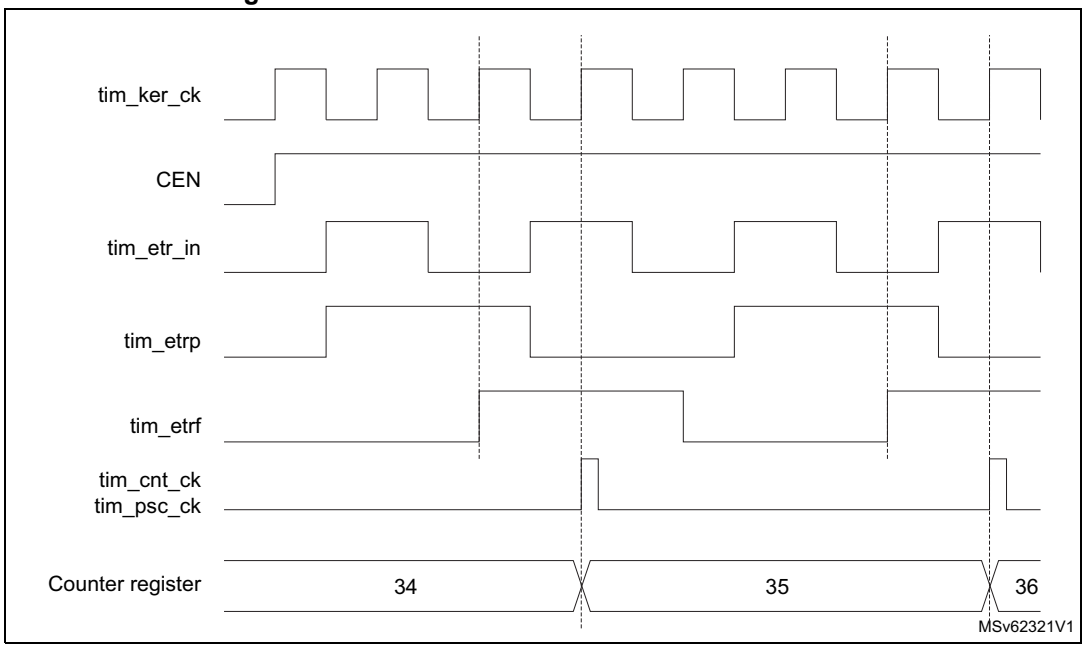
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 450. Control circuit in external clock mode 2



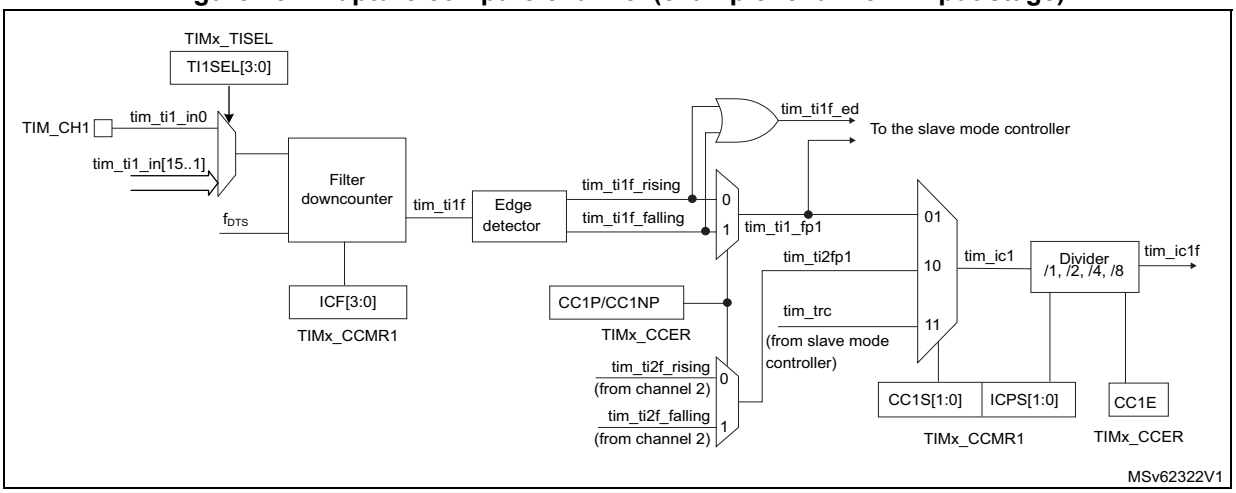
35.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 451 to *Figure 454* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 451. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 452. Capture/compare channel 1 main circuit

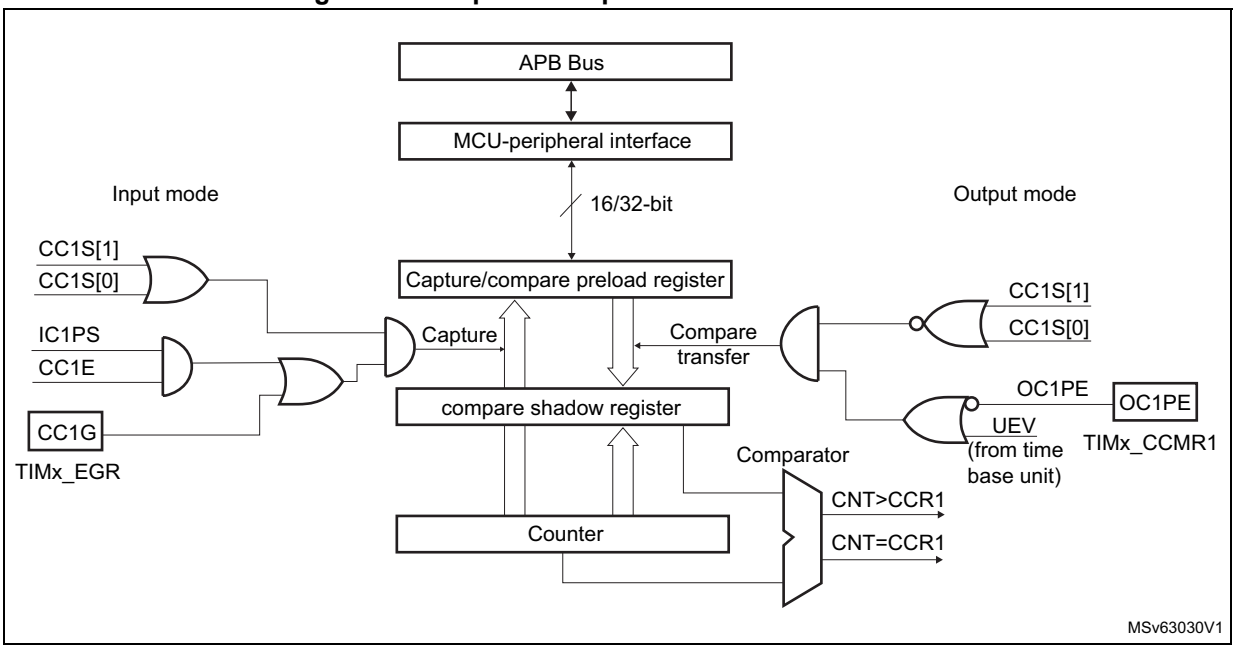


Figure 453. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)

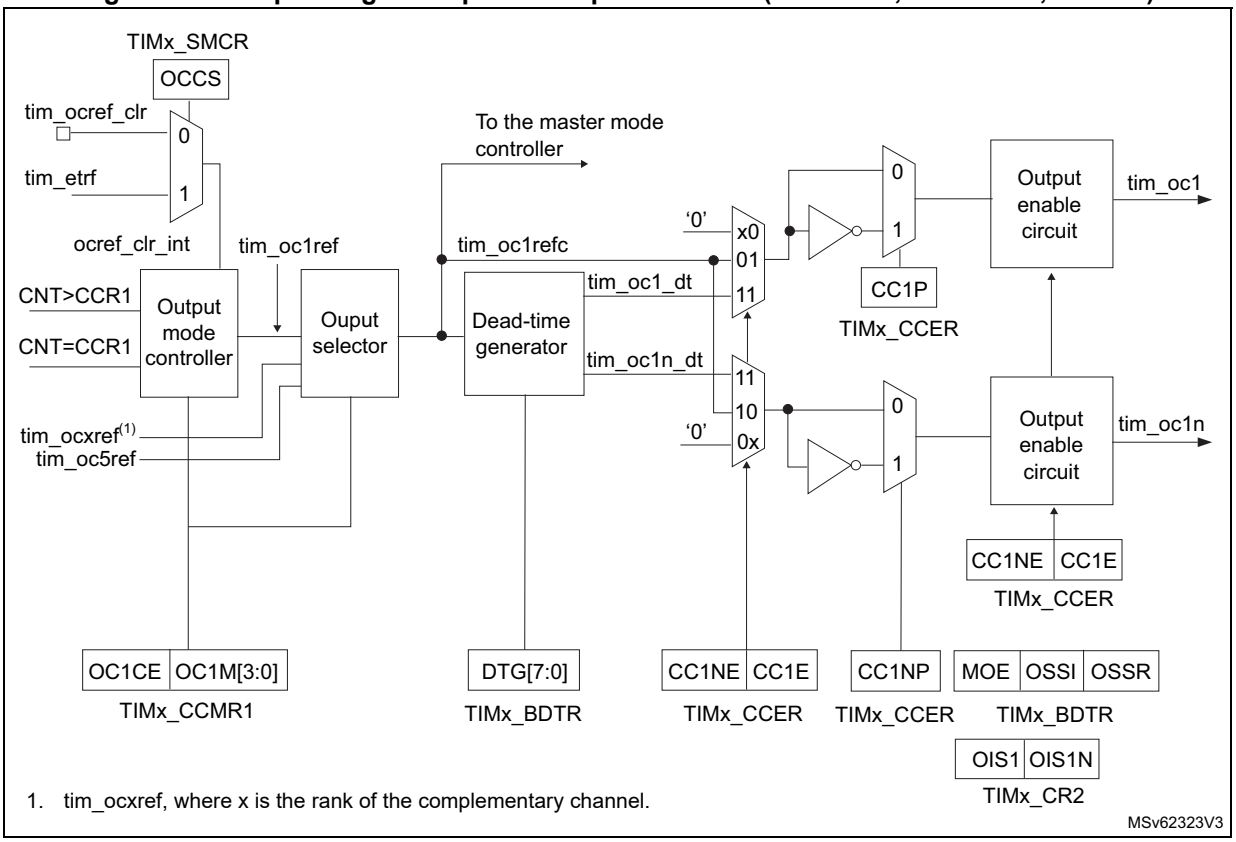
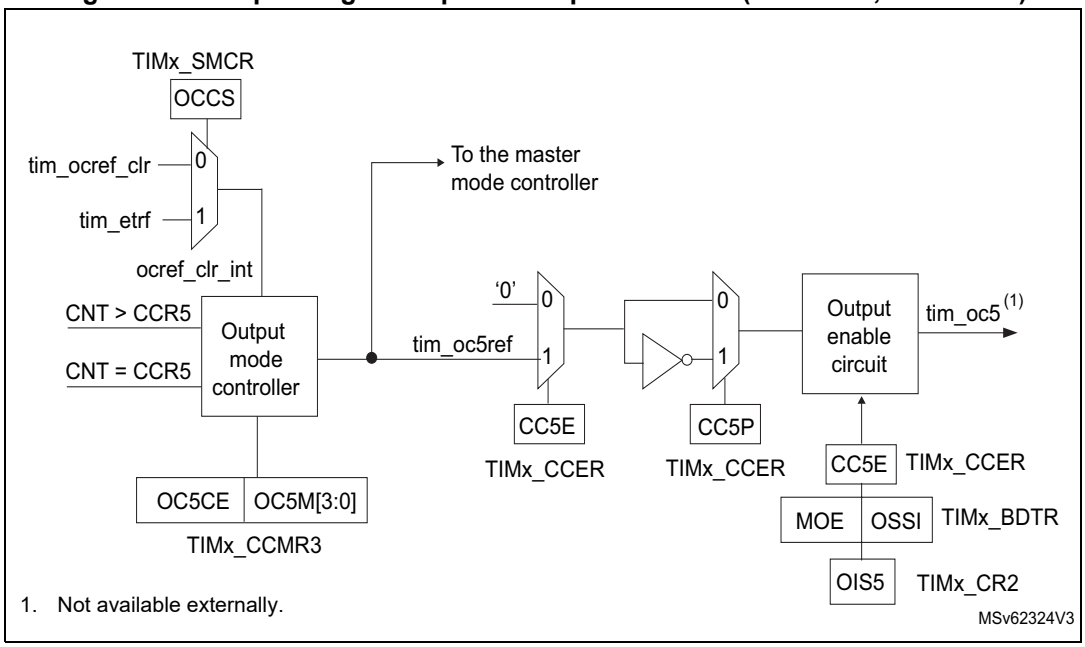


Figure 454. Output stage of capture/compare channel (channel 5, idem ch. 6)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

35.3.9 Input capture mode

In Input capture mode, the capture/compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_ti1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

35.3.10 PWM input mode

This mode allows to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges)

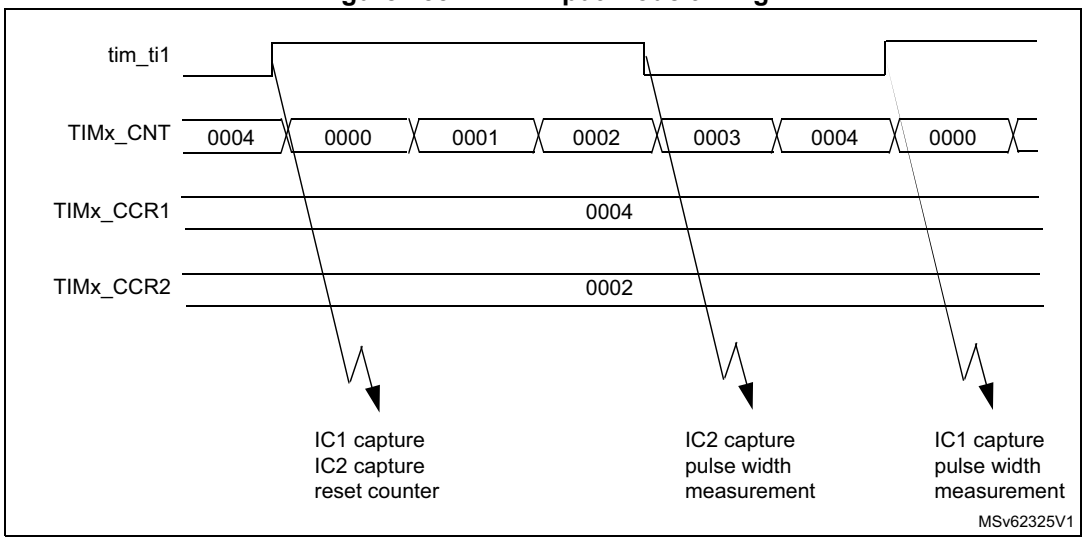
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input
- These 2 ICx signals are active on edges with opposite polarity
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 455. PWM input mode timing



Note: The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals.

35.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

35.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

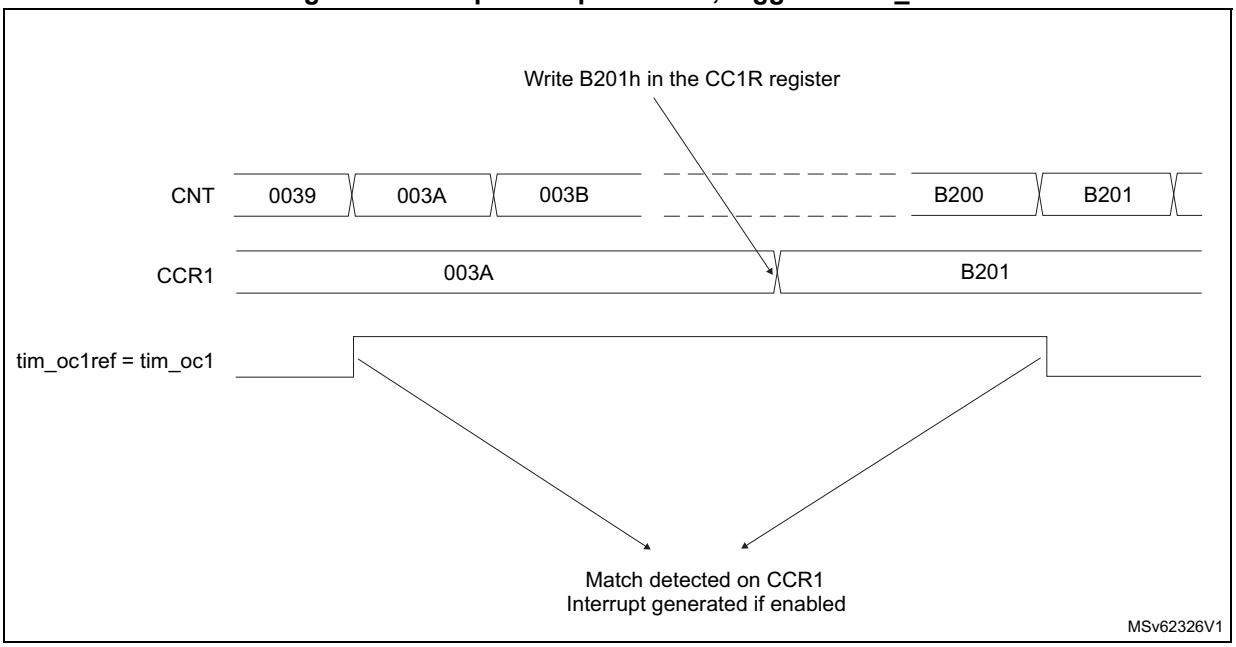
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 456*.

Figure 456. Output compare mode, toggle on tim_oc1



35.3.13 PWM mode

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

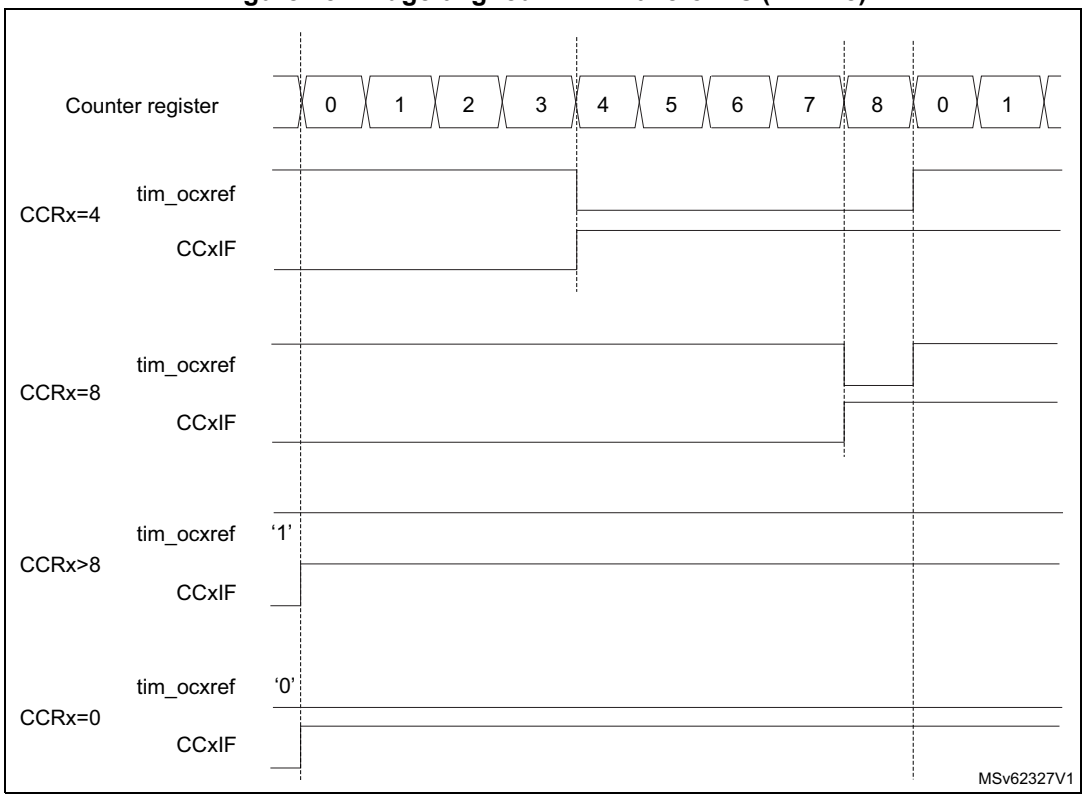
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the section *Upcounting mode*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then tim_ocxref is held at ‘1’. If the compare value is 0 then tim_ocxref is held at ‘0’. *Figure 457* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 457. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the section *Downcounting mode*.

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then tim_ocxref is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

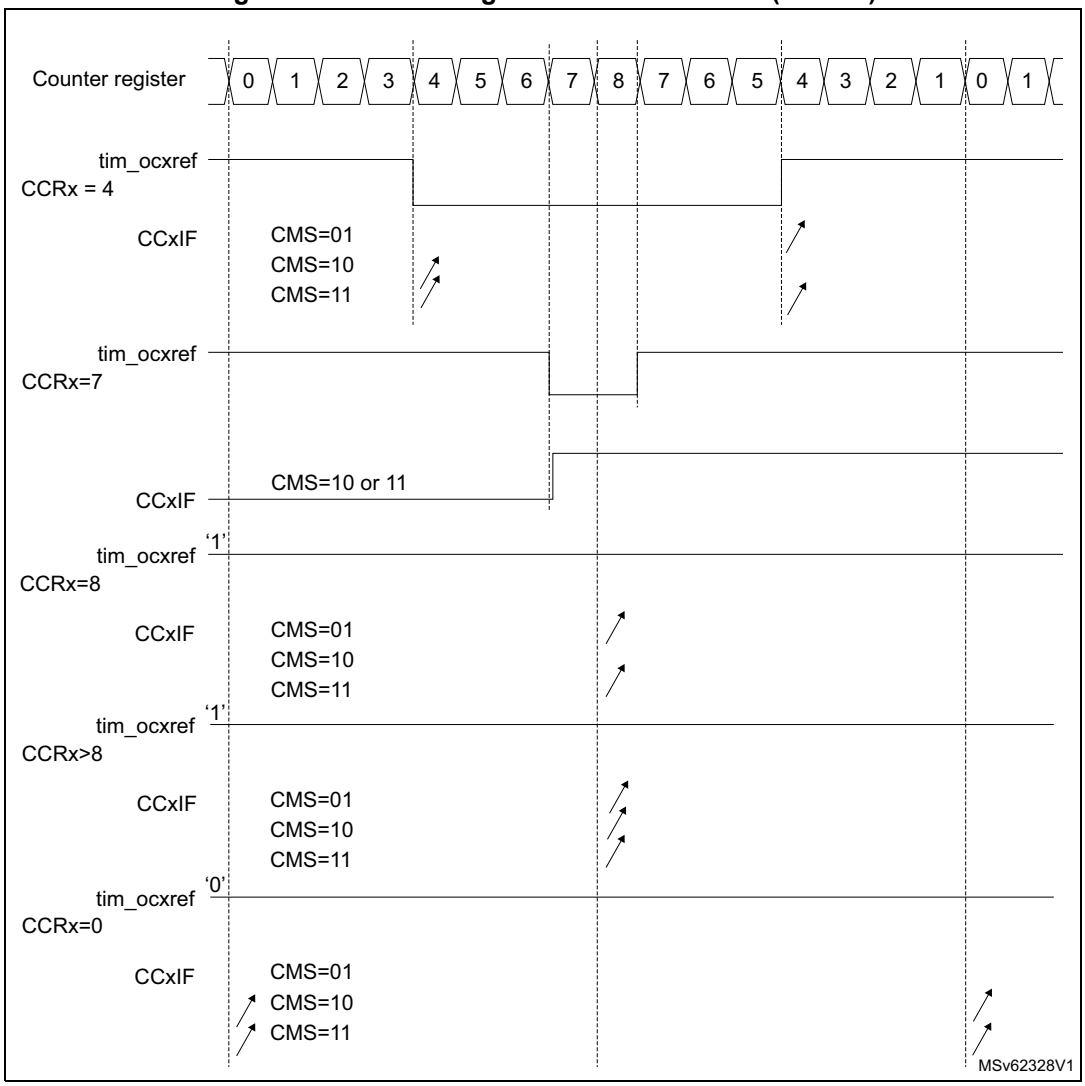
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the section *Center-aligned mode (up/down counting)*.

Figure 458 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register

Figure 458. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

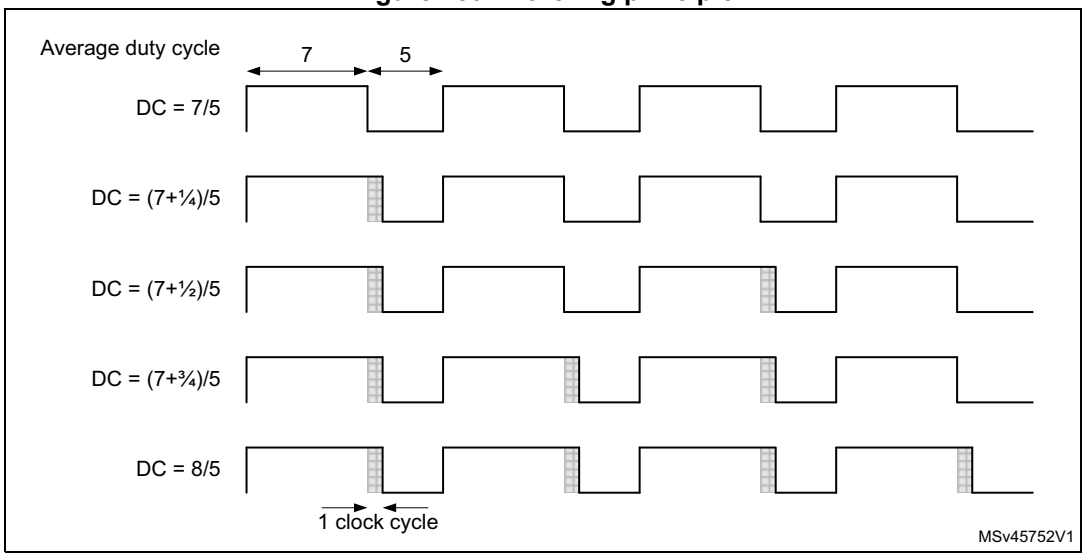
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 459* presents the dithering principle applied to 4 consecutive PWM cycles.

Figure 459. Dithering principle



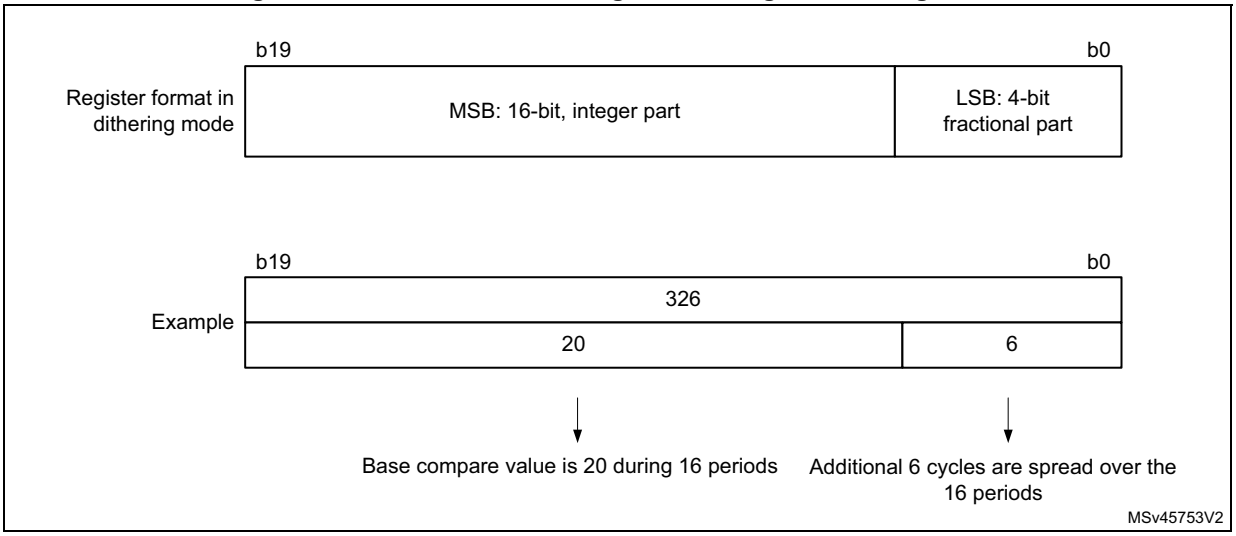
When the dithering mode is enabled, the register coding is changed as follows (refer to *Figure 460* for example):

- The 4 LSBs are coding for the enhanced resolution part (fractional part)
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value

Note: The ARR and CCR values are updated automatically if the DITHEN bit is set / reset (for instance, if ARR= 0x05 with DITHEN=0, it is updated to ARR = 0x50 with DITHEN=1).
The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The DITHEN bit must be reset

4. *The CCIF flags must be cleared*
5. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 460. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

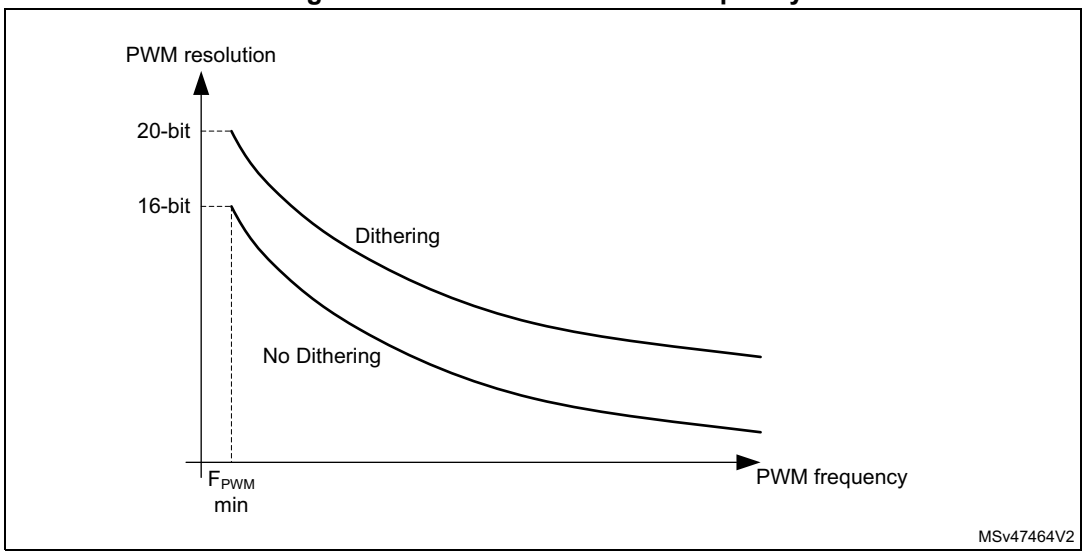
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

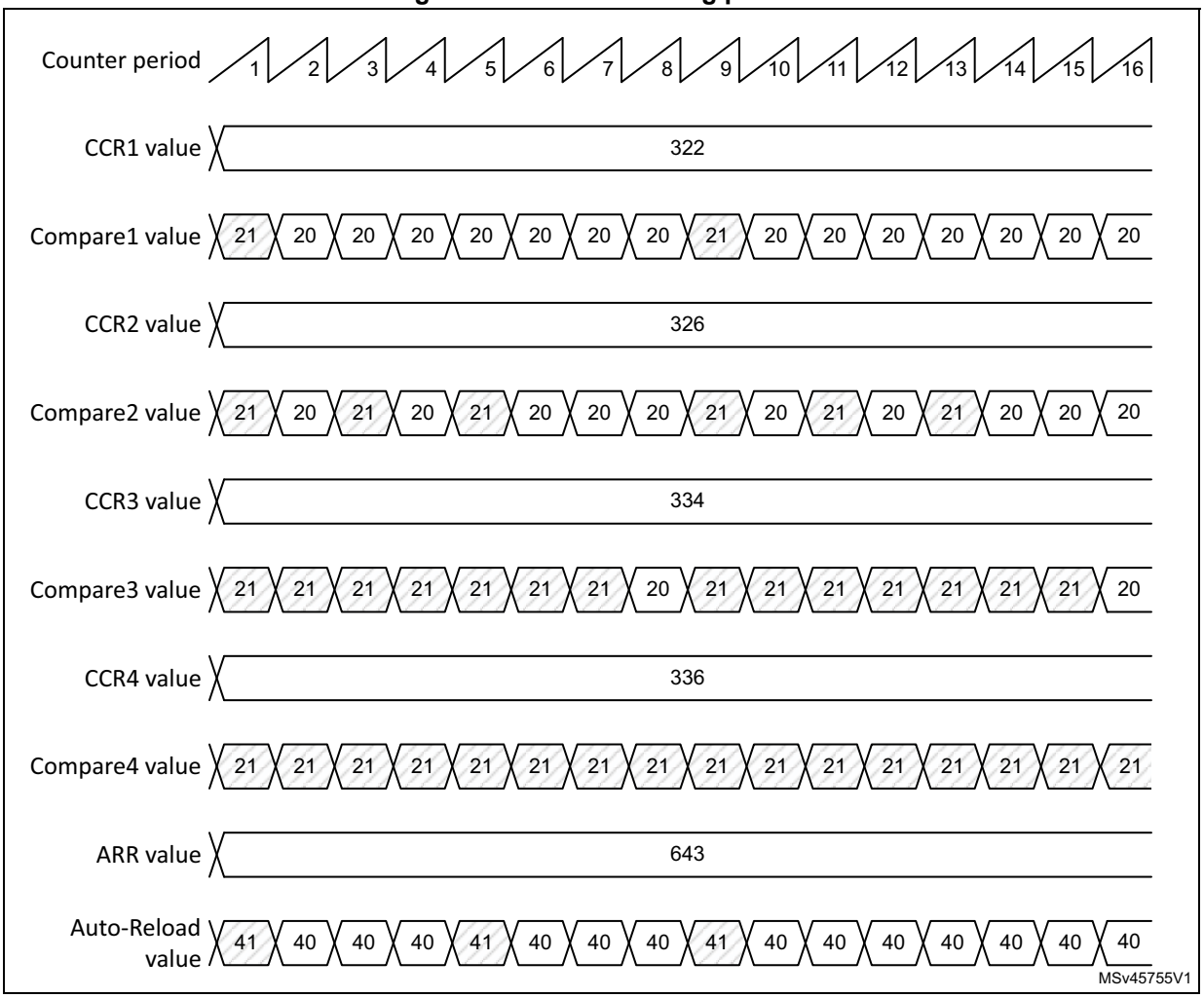
As shown in *Figure 461*, the dithering mode allows to increase the PWM resolution whatever the PWM frequency.

Figure 461. PWM resolution vs frequency



The duty cycle and / or period changes are spread over 16 consecutive periods, as described in *Figure 462*.

Figure 462. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in *Table 412*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 412. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 412. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to '00'). In this case, the dithering pattern is applied over 8 consecutive PWM periods, considering the up and down counting phases as shown in *Figure 463*.

Figure 463. Dithering effect on duty cycle in center-aligned PWM mode

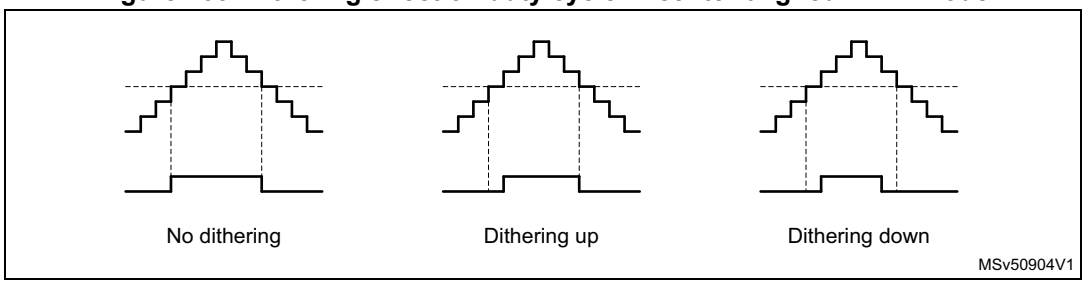


Table 413 shows how the dithering pattern is added in center-aligned PWM mode.

Table 413. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-

Table 413. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

35.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

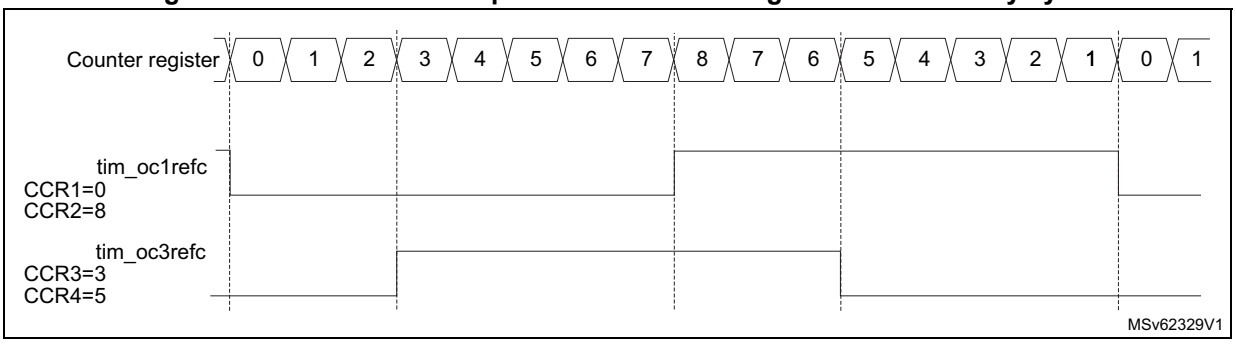
Asymmetric PWM mode can be selected independently for two channels (one tim_ocx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if a tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or a tim_oc2refc signal resulting from asymmetric PWM mode 1.

Figure 464 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 464. Generation of 2 phase-shifted PWM signals with 50% duty cycle



35.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

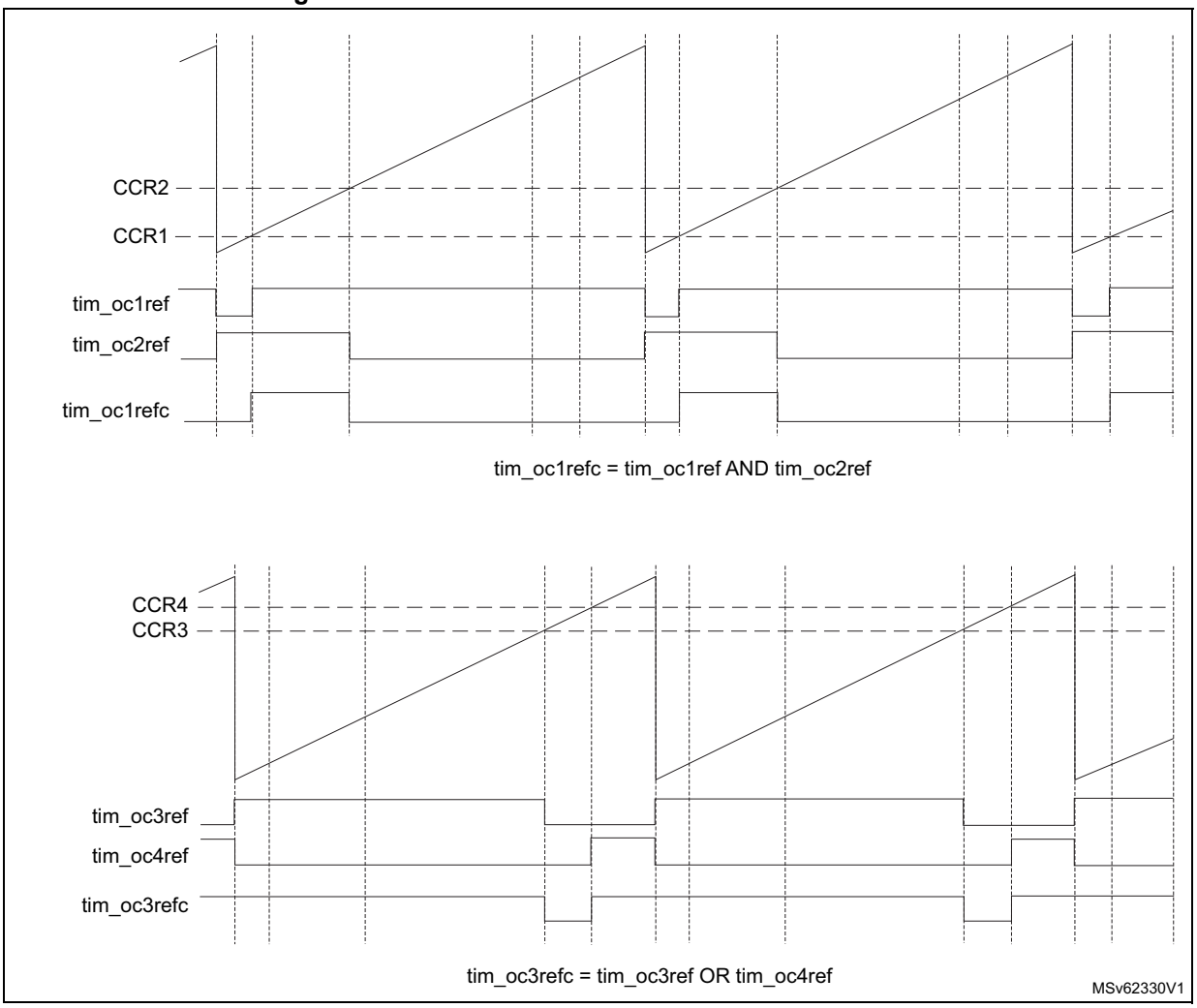
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 465 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2
- Channel 2 is configured in PWM mode 1
- Channel 3 is configured in Combined PWM mode 1
- Channel 4 is configured in PWM mode 2

Figure 465. Combined PWM mode on channel 1 and 3



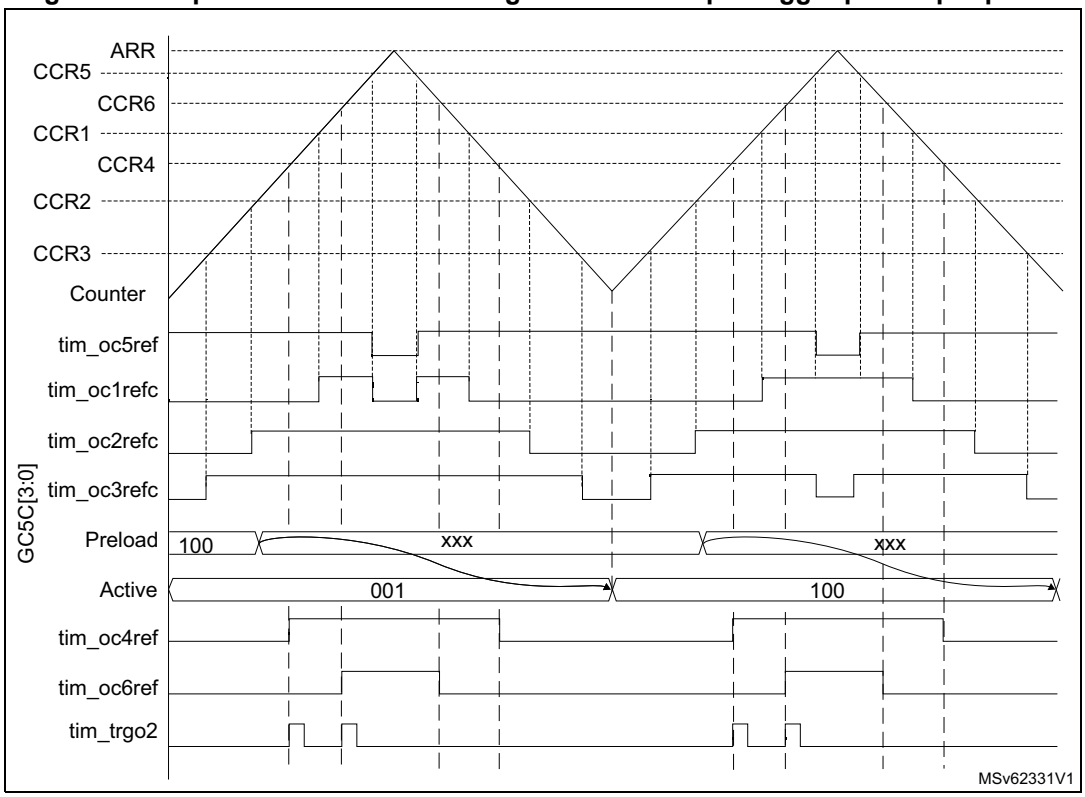
35.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, tim_oc3refc is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bit GC5C[3:1].

Figure 466. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Please refer to *Section 35.3.31: ADC synchronization* for more details.

35.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...).

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 422* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

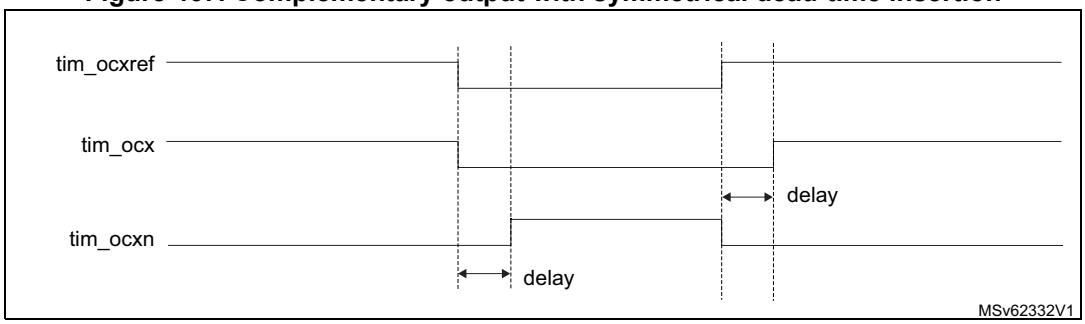
reference waveform tim_ocxref, it generates 2 outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 467. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIMx_DTR2 allows to differentiate the deadtime values for rising and falling edges of the reference signal, as shown in *Figure 468*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bit field in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bit field in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN=1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bit field DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 468. Asymmetrical deadtime

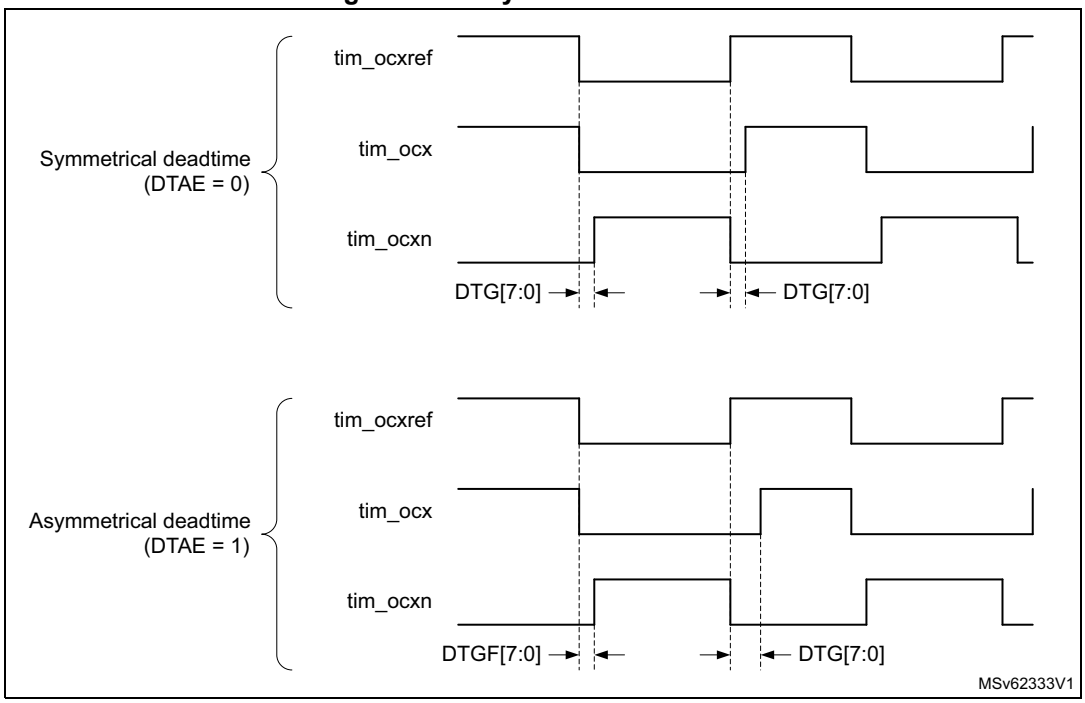


Figure 469. Dead-time waveforms with delay greater than the negative pulse

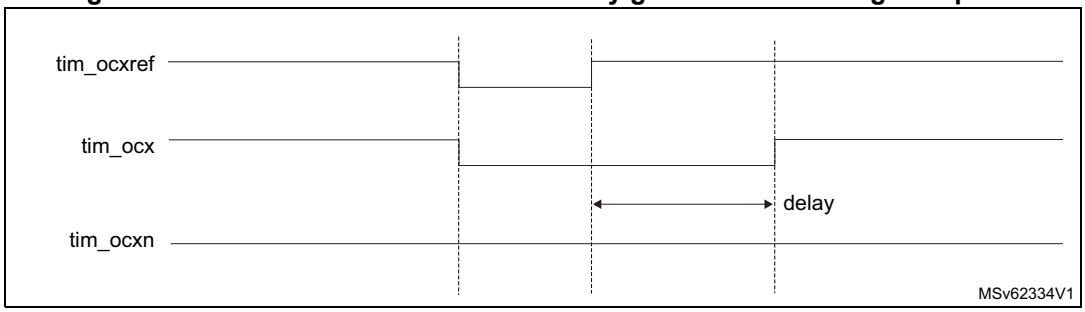
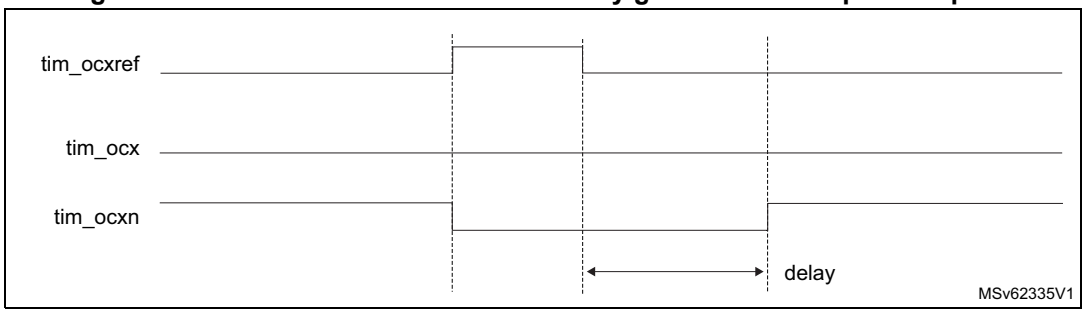


Figure 470. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 35.6.21: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)* for delay calculation.

Re-directing tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be re-directed to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP=0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE=CCxNE=1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

35.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 422: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous

and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pins (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation).
  - coming from a system break request (refer to *Section 35.3.2* for product specific implementation).

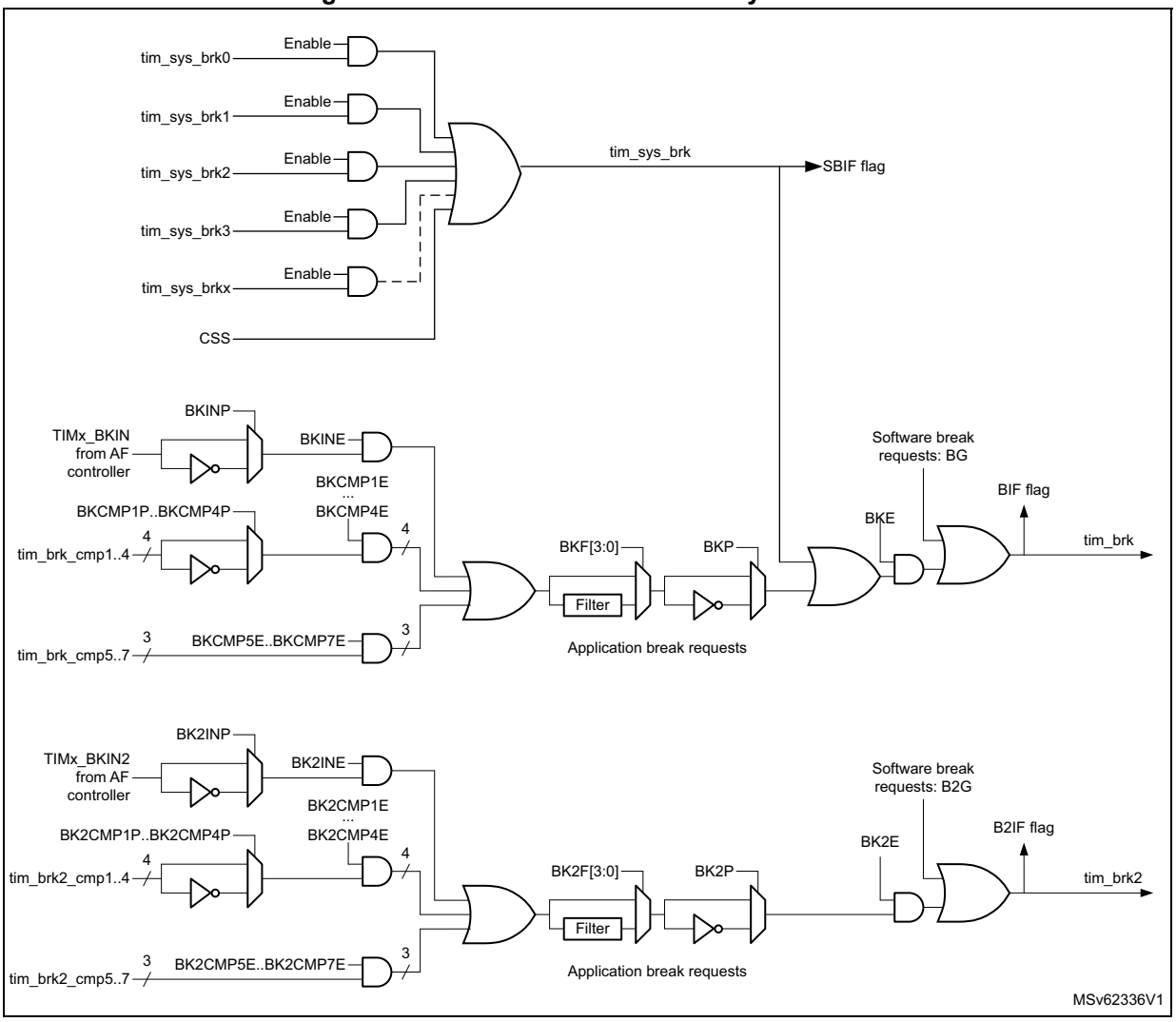
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pins (as per selection done in the AFIO controller), with polarity selection and optional digital filtering.
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 35.3.2* for product specific implementation).

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 471*.

Figure 471. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

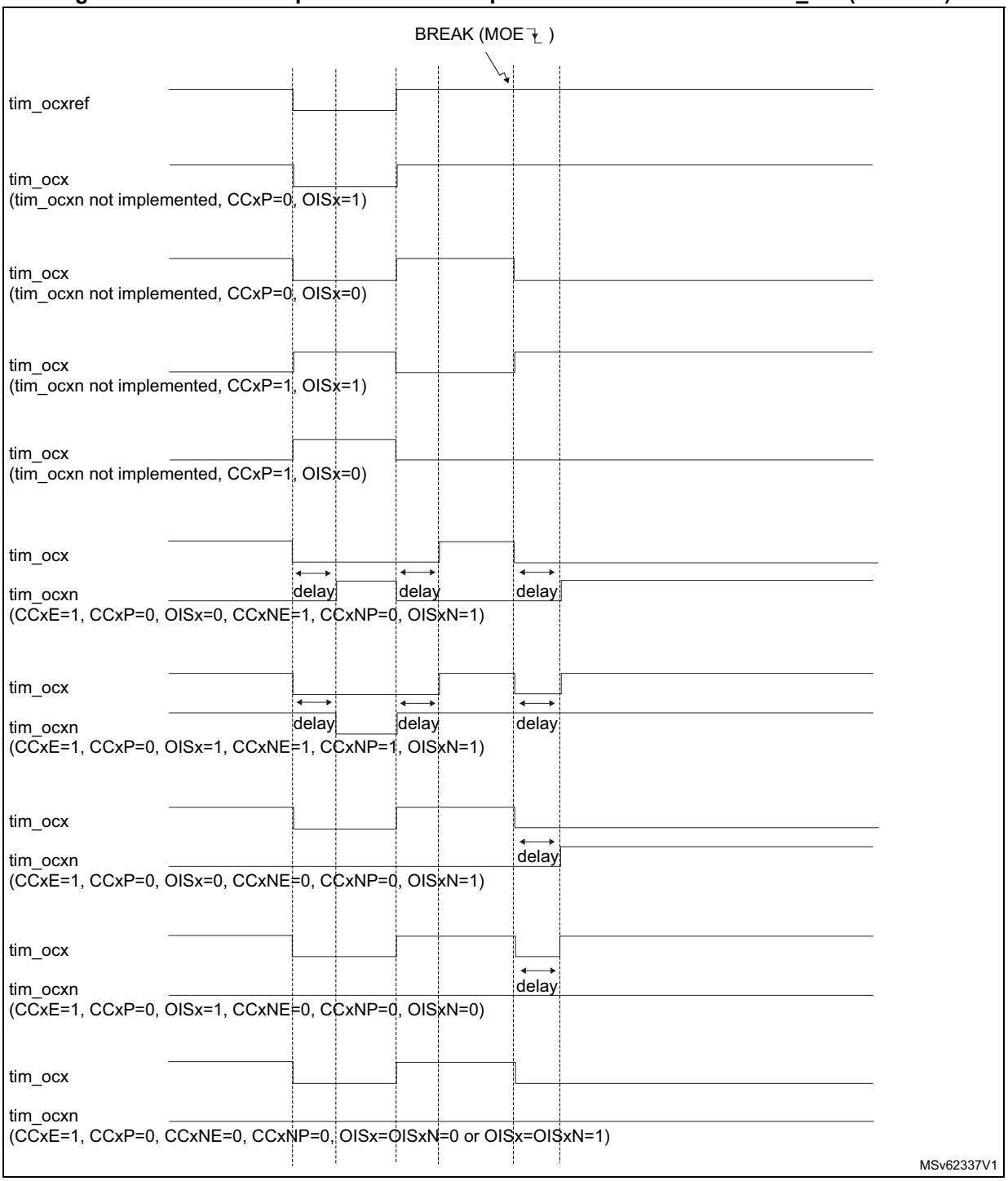
- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 35.6.21: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)*. The LOCK bits can be written only once after an MCU reset.

Figure 472 shows an example of behavior of the outputs in response to a break.

Figure 472. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 414*.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 414. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (Low side switches)	tim_ocx output (High side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 473 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 473. PWM output state following tim_brk and tim_brk2 assertion (OSSI=1)

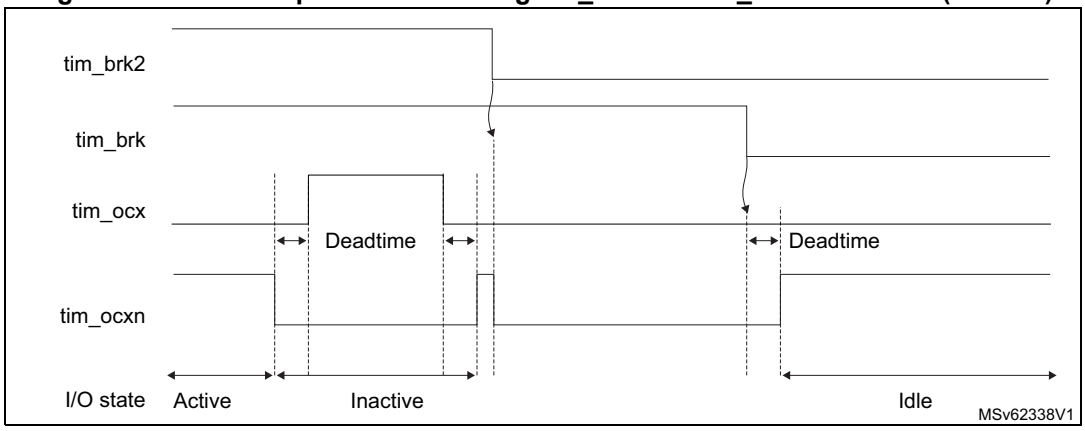
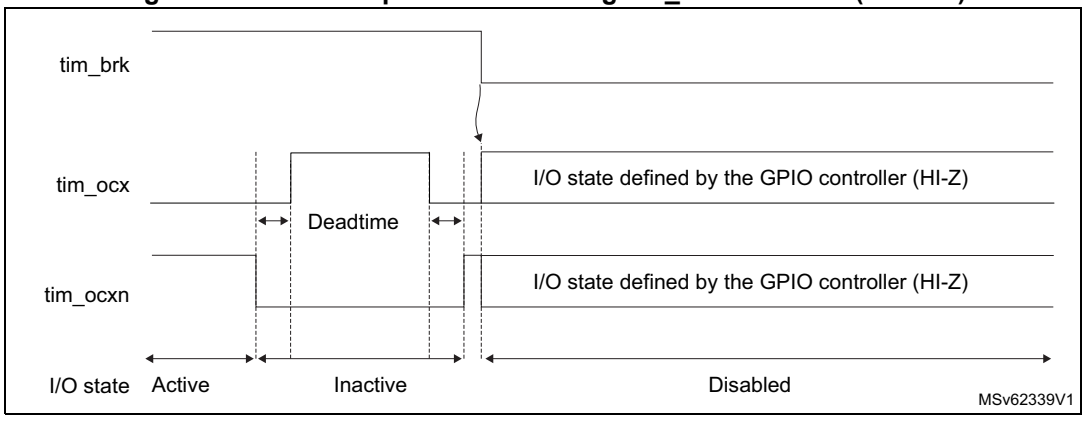


Figure 474. PWM output state following tim_brk assertion (OSSI=0)



35.3.19 Bidirectional break inputs

The TIM1/TIM8 are featuring bidirectional break I/Os, as represented in *Figure 475*.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and requires the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (for example, CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to '0' to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKIN and TIMx_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output from being re-started as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (refer to *Table 415*).

Table 415. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

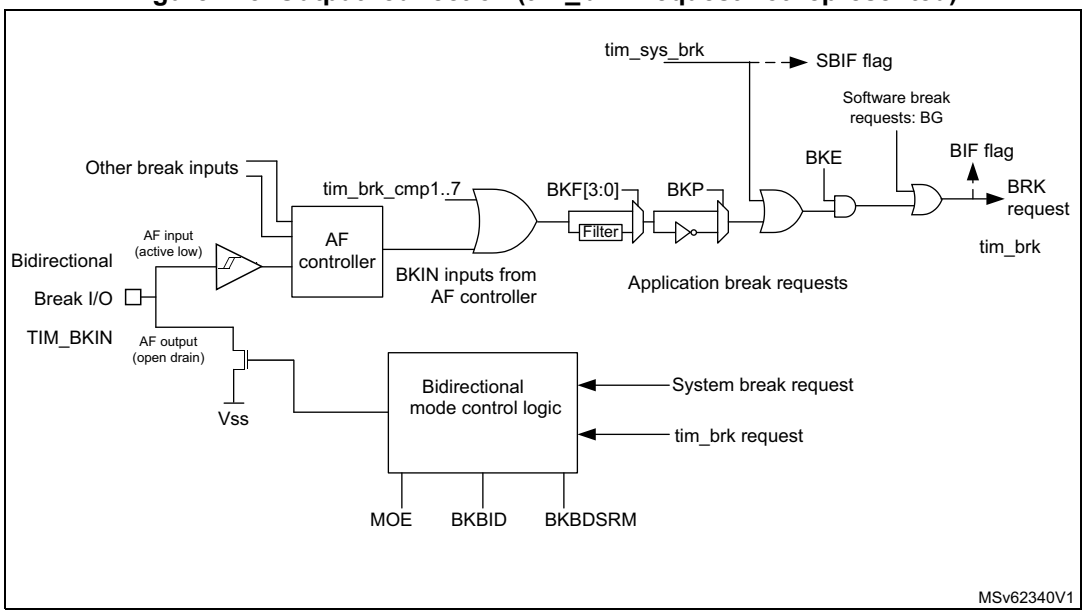
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 475. Output redirection (tim_brk2 request not represented)

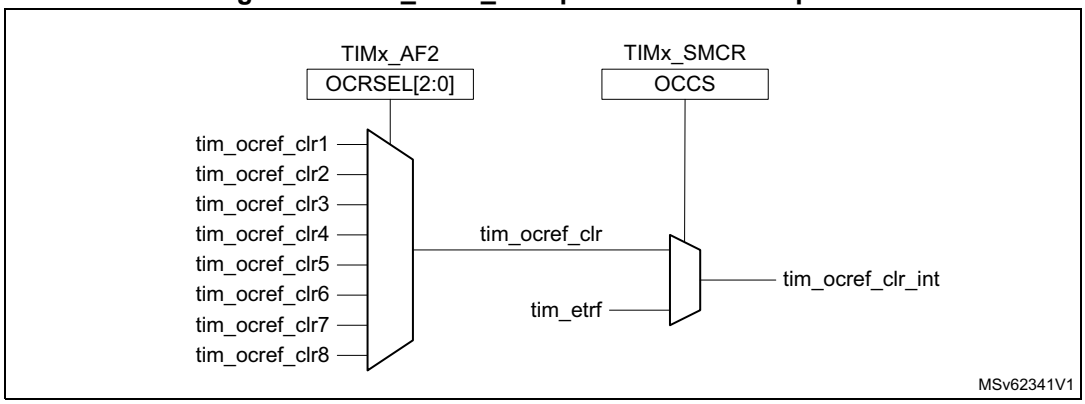


35.3.20 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.
tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

The tim_ocref_clr input can be selected among several inputs, using the OCRSEL[2:0] bit field in the TIMx_AF2 register, as shown in *Figure 476*. Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals* for a list of sources available in the product.

Figure 476. tim_ocref_clr input selection multiplexer

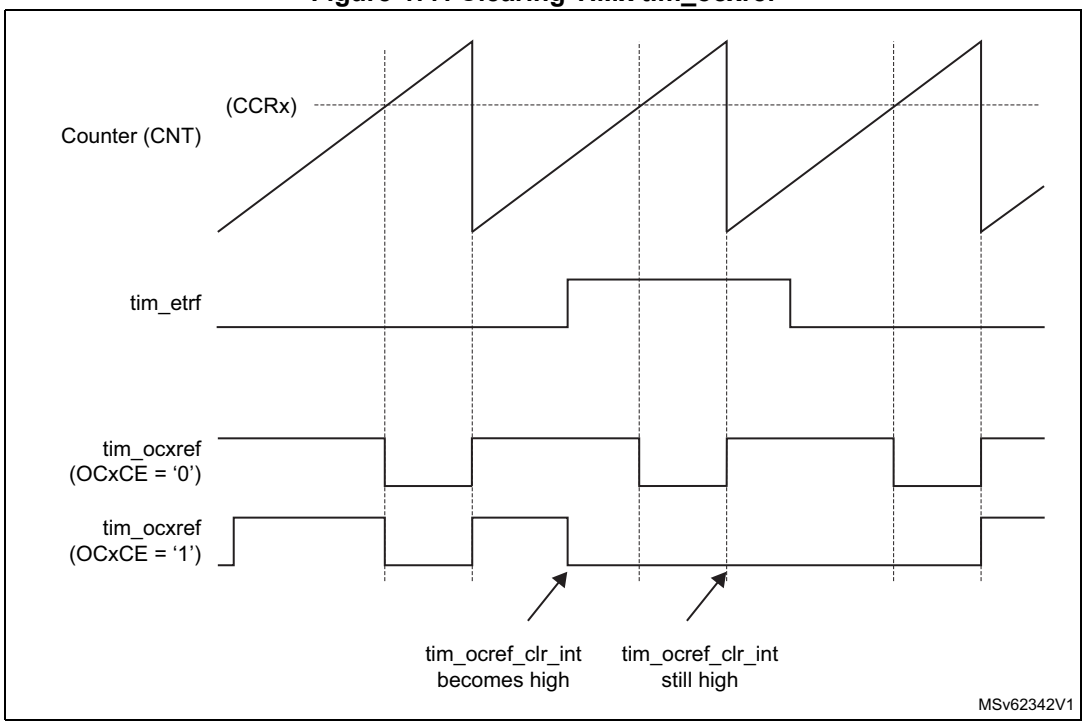


When tim_etrf is chosen, tim_etr_in must be configured as follows:

1. The External Trigger Prescaler must be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 477 shows the behavior of the tim_ocxref signal when the tim_etrf Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 477. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx > ARR), then tim_ocxref is enabled again at the next counter overflow.

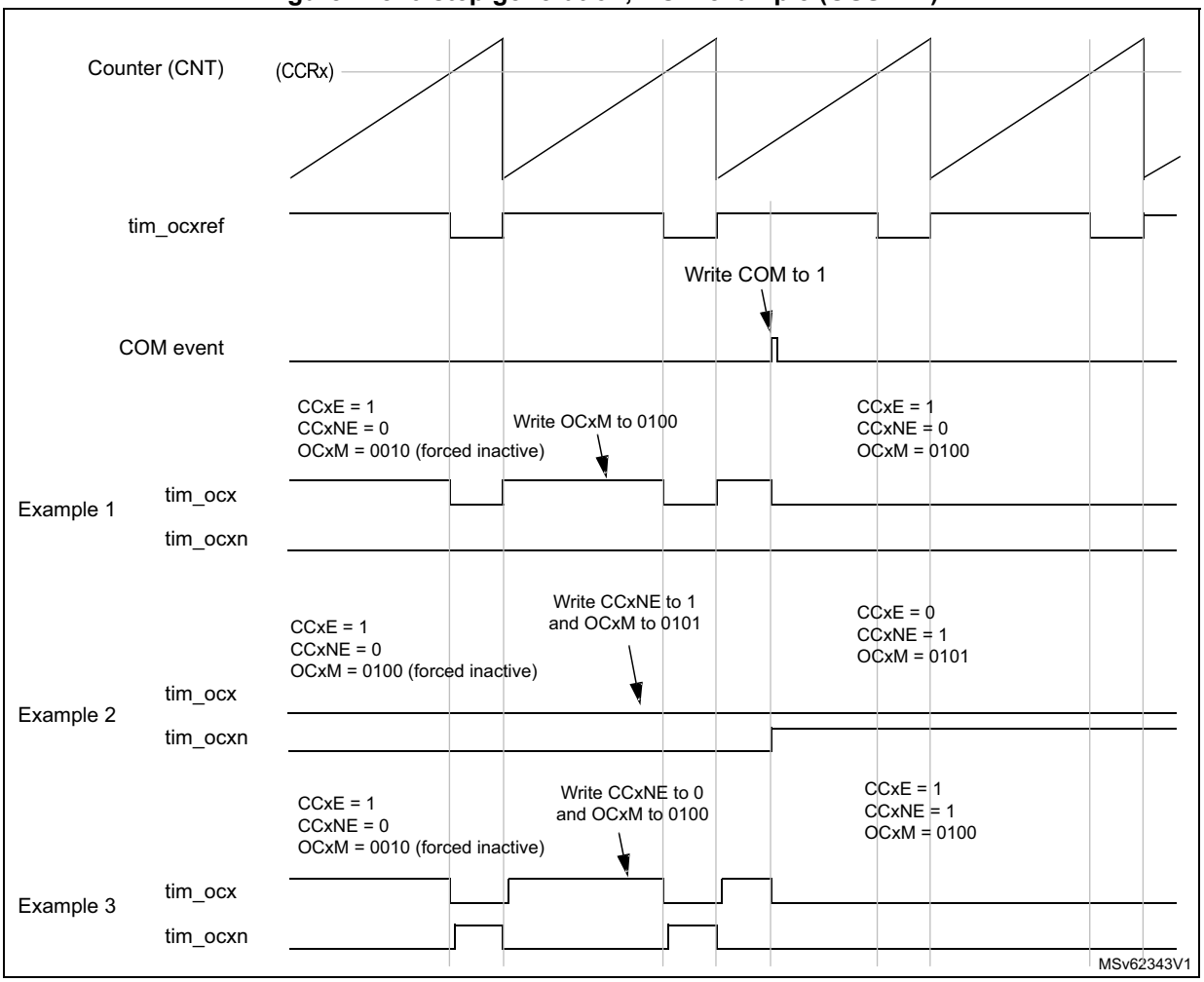
35.3.21 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

Figure 478 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 478. 6-step generation, COM example (OSSR=1)



35.3.22 One-pulse mode

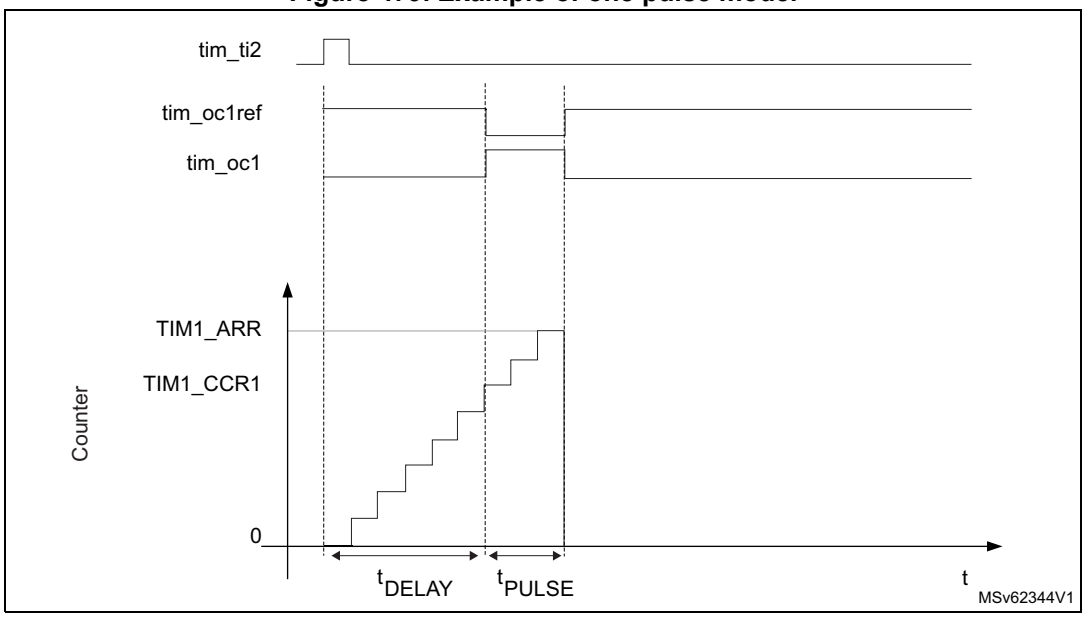
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 479. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S='01' in the TIMx_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS=00110 in the TIMx_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one wants to build a waveform with a transition from '0' to '1' when a compare match occurs, and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this, PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally, the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

35.3.23 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 35.3.22: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

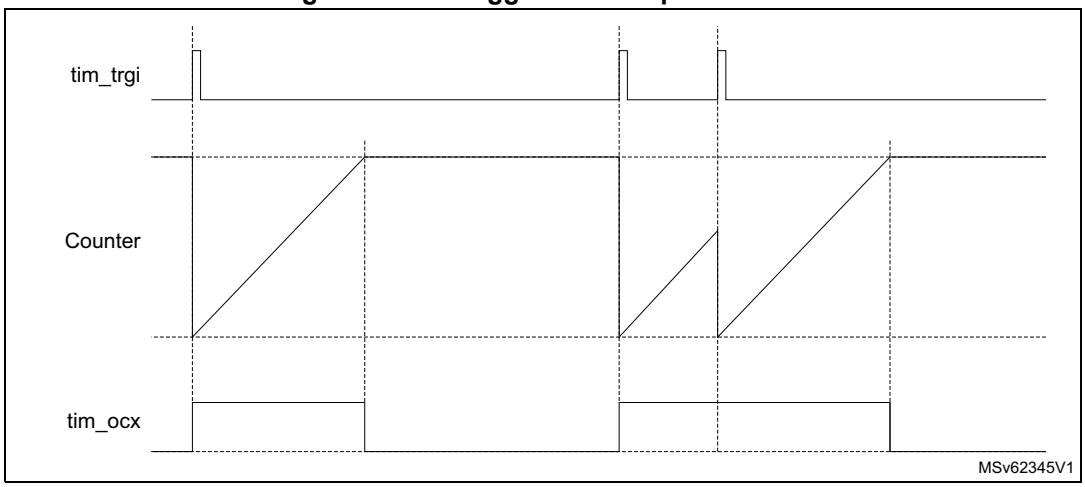
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bits are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 480. Retriggerable one pulse mode



35.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown in *Figure 481*.

Figure 481. Pulse generator circuitry

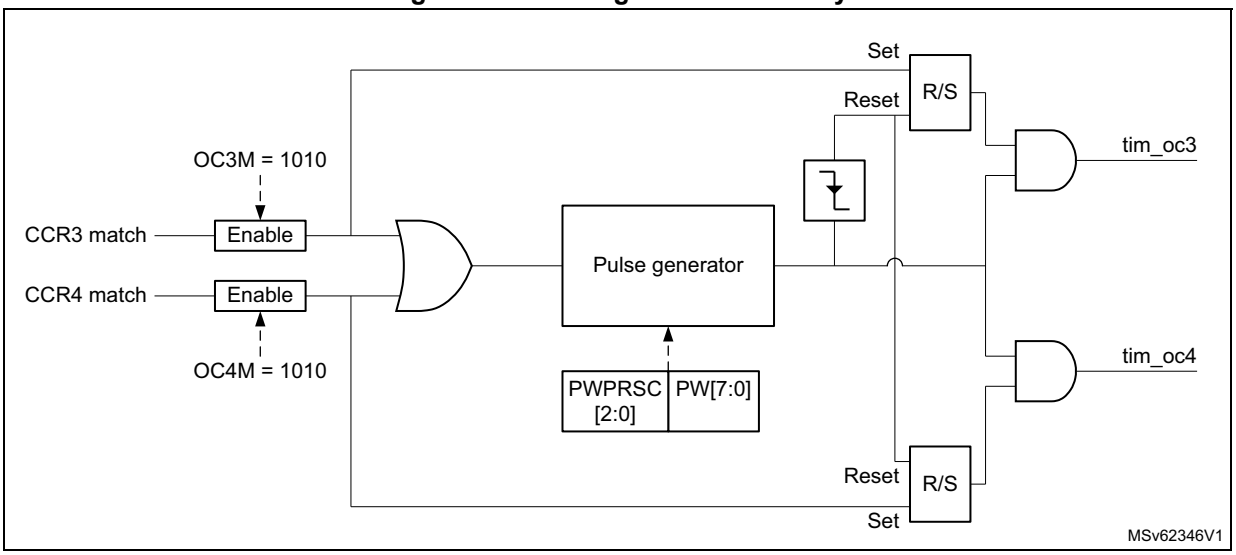
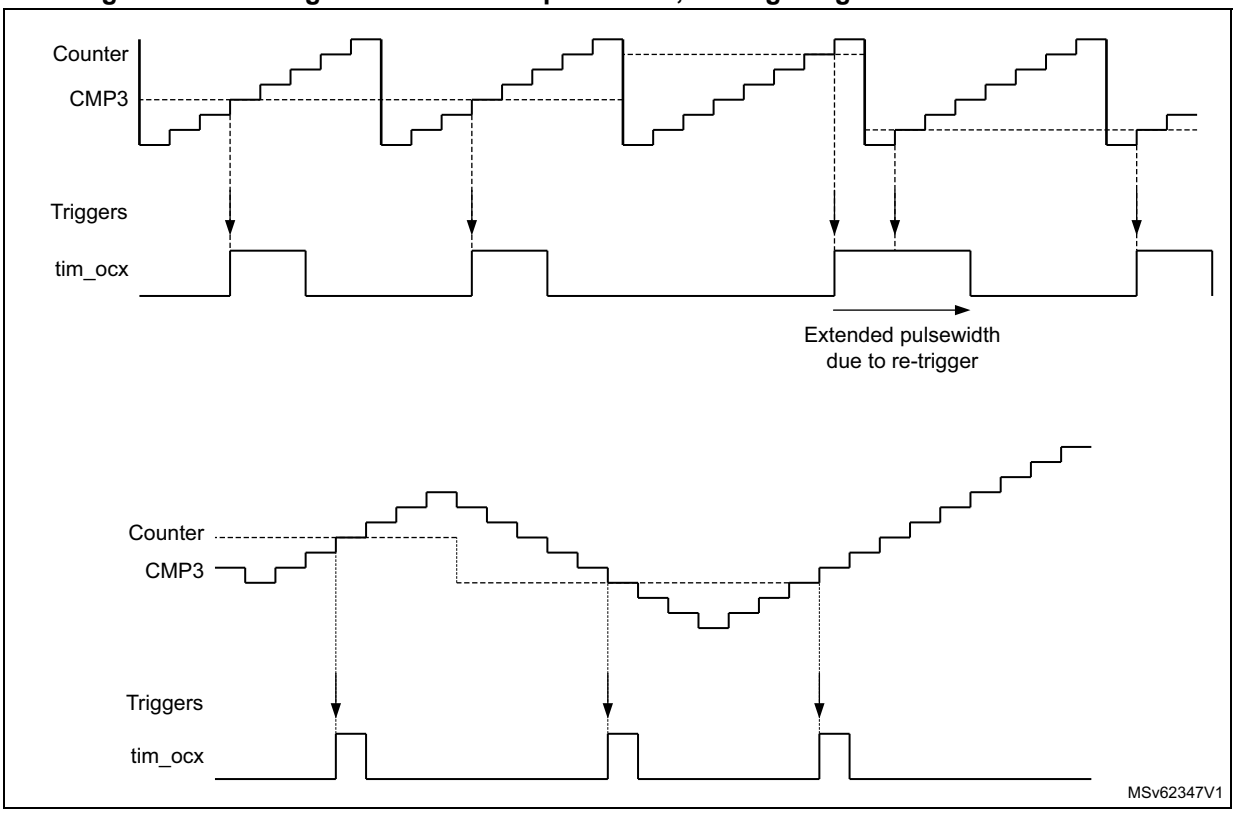


Figure 482 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 482. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bit fields in TIMx_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bit field in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

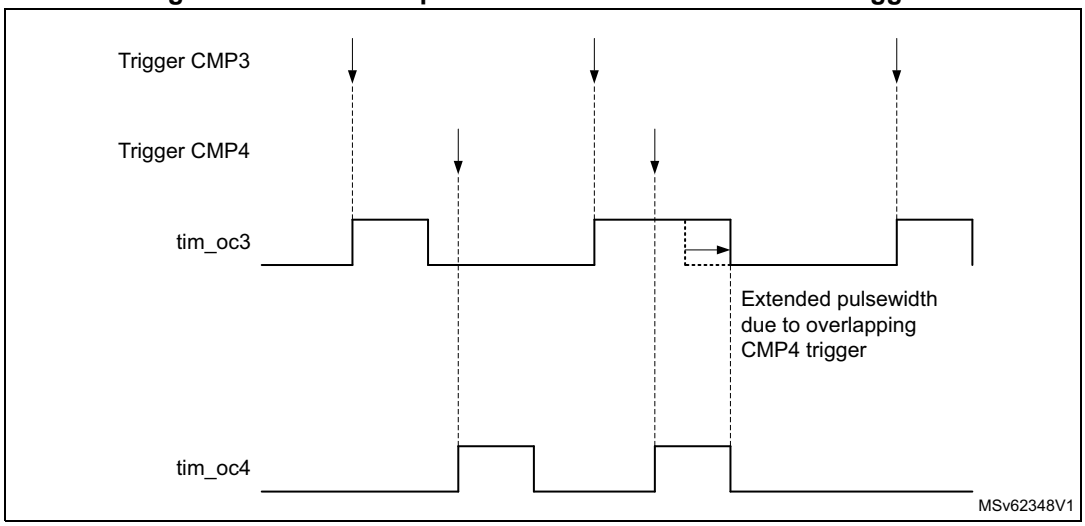
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is on-going causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the 1st arriving trigger is extended (because of the re-trigger), while the pulse width of the last arriving trigger is correct (as shown in the Figure 483).

Figure 483. Extended pulsewidth in case of concurrent triggers



35.3.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS='0001' in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS='0010' if it is counting on tim_ti2 edges only and SMS='0011' if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to a quadrature encoder. Refer to *Table 416*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1=tim_ti1 if not filtered and not inverted, tim_ti2fp2=tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 416. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output, which indicates the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 484 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1)
- CC2S='01' (TIMx_CCMR2 register, tim_ti1fp2 mapped on tim_ti2)
- CC1P='0' and CC1NP='0' (TIMx_CCER register, tim_ti1fp1 non-inverted, tim_ti1fp1=tim_ti1)
- CC2P='0' and CC2NP='0' (TIMx_CCER register, tim_ti1fp2 non-inverted, tim_ti1fp2= tim_ti2)
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN='1' (TIMx_CR1 register, Counter enabled)

Figure 484. Example of counter operation in encoder interface mode.

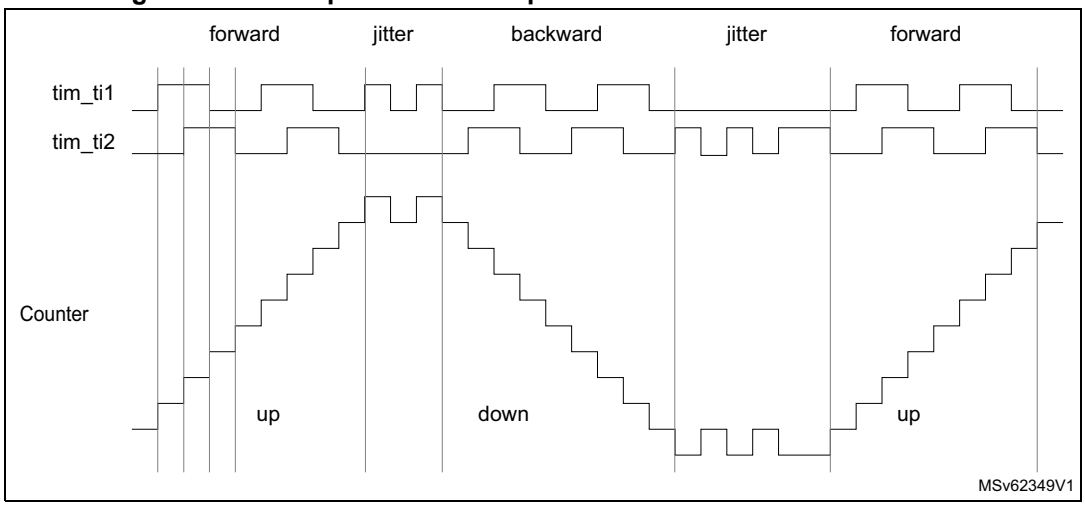


Figure 485 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P='1').

Figure 485. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

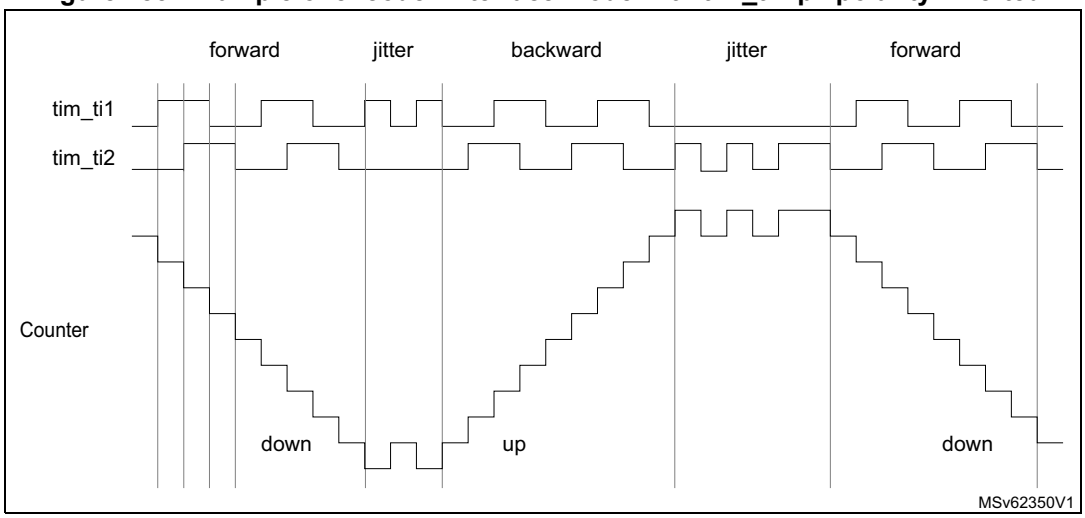
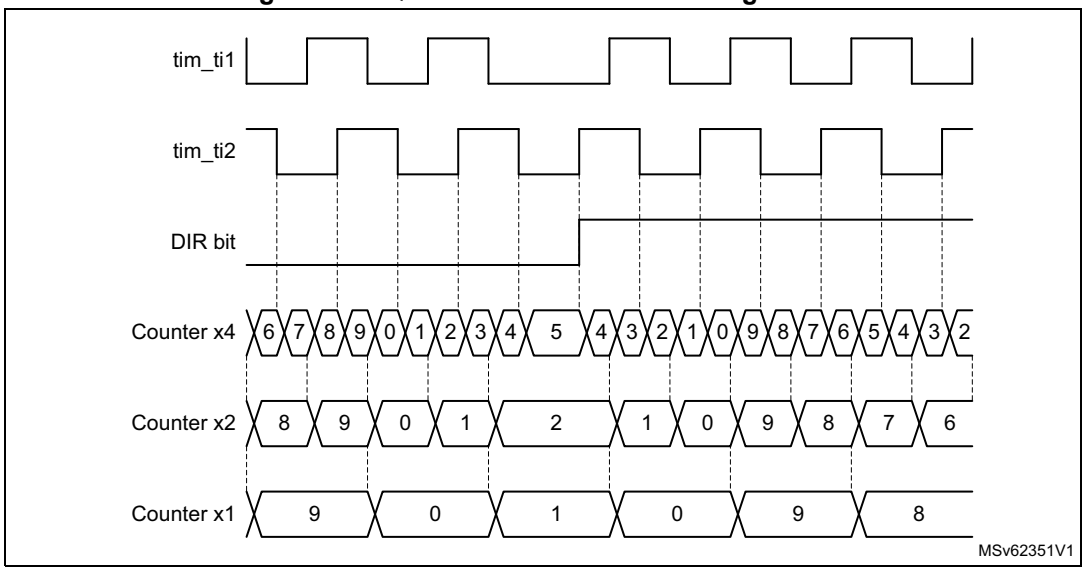


Figure 486 shows the timer counter value during a speed reversal, for various counting modes.

Figure 486. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

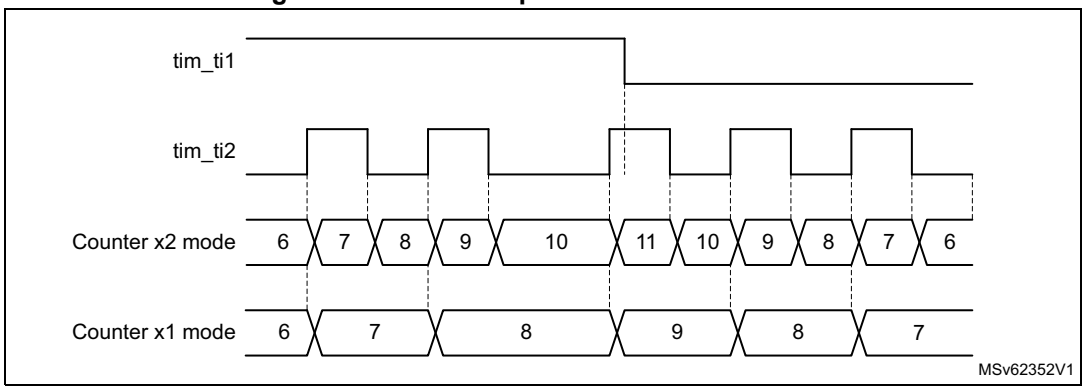
In the "clock plus direction" mode shown in *Figure 487*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bit field in the TIMx_SMCR register, as follows:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 487. Direction plus clock encoder mode



Directional Clock encoder mode

In the “directional clock” mode in *Figure 488*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bit field in the TIMx_SMCR register, as follows:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 488*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 489*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 488*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 489*).

Figure 488. Directional clock encoder mode (CC1P = CC2P = 0)

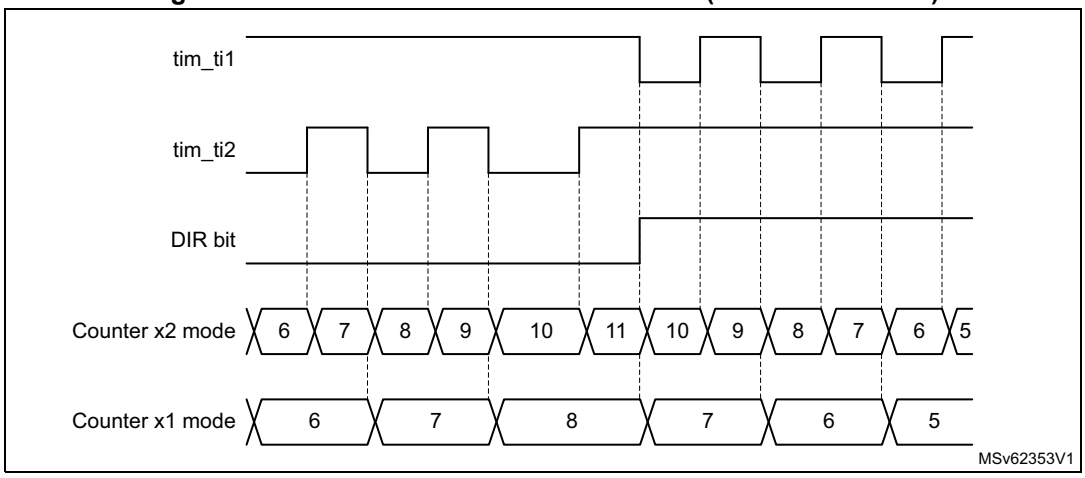


Figure 489. Directional clock encoder mode (CC1P = CC2P = 1)

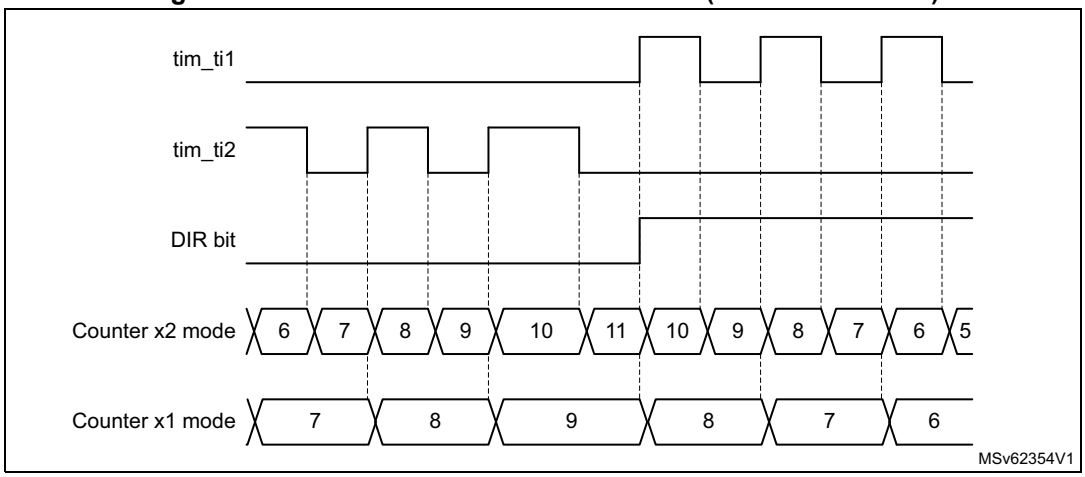


Table 417 details how the directional clock mode operates, for any input transition.

Table 417. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP=0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP=1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP=0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP=1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index Input

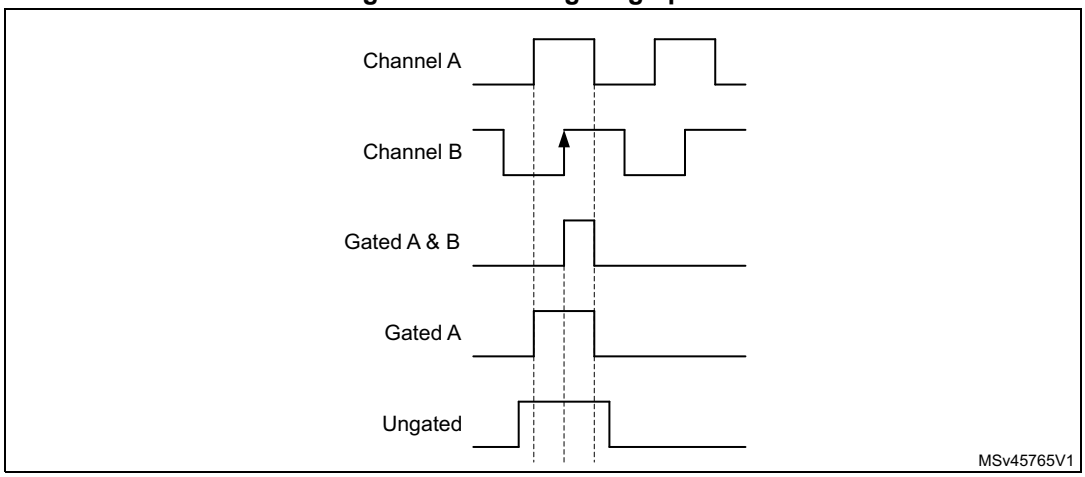
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The Index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMX_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bit field has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Commercially available encoders are proposed with several options for index pulse conditioning, as per *Figure 490*:

- Gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- Gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- Ungated: the pulsewidth is up to one channel period, without any alignment to the edges

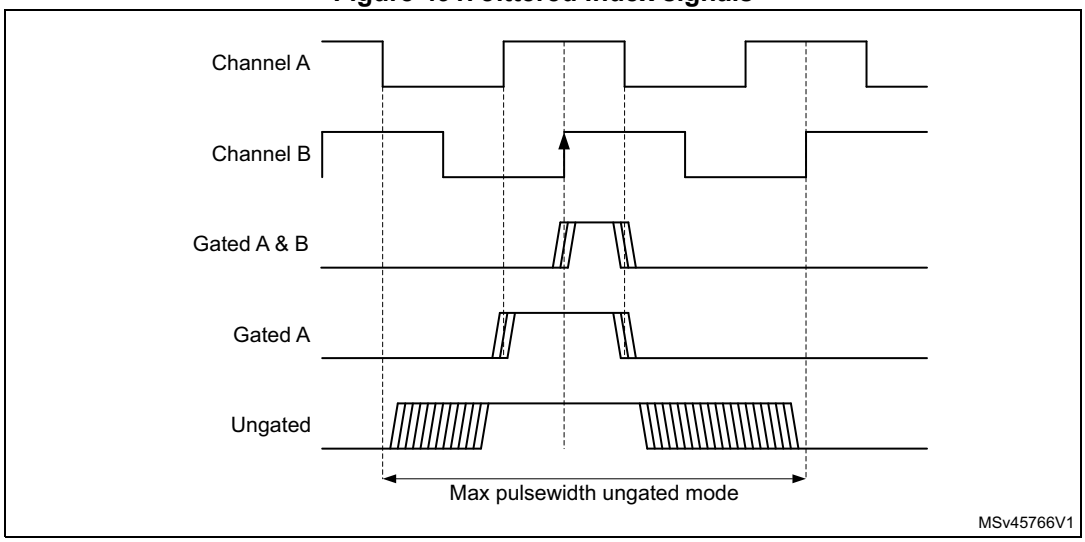
Figure 490. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown in *Figure 491*.

In ungated mode, the signal must be strictly below 2 encoder periods. If the pulsewidth is greater than or equal to 2 encoder periods, the counter is reset multiple times.

Figure 491. Jittered Index signals



The timer supports the 3 gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (that are channel A and

channel B state combination) the index must be synchronized, using the IPOS[1:0] bit field in the TIMx_ECR register.

The Index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0)
- The counter is set to TIMx_ARR when down counting

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 492* shows at which position is the index generated, for a simplistic example (an encoder providing 4 edges par mechanical rotation).

Figure 492. Index generation for IPOS[1:0] = 11

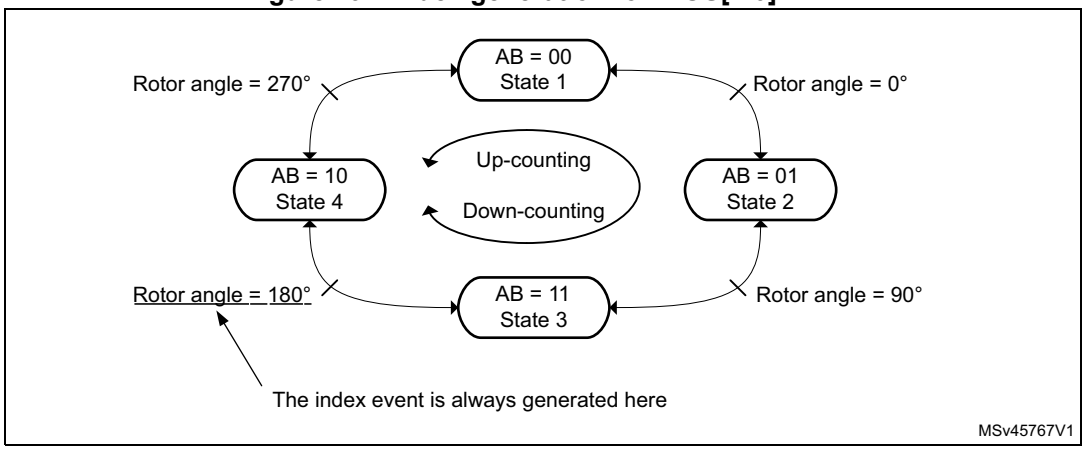


Figure 493 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is '11' (ChA=1, ChB=1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the '11' state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 493. Counter reading with index gated on channel A (IPOS[1:0] = 11)

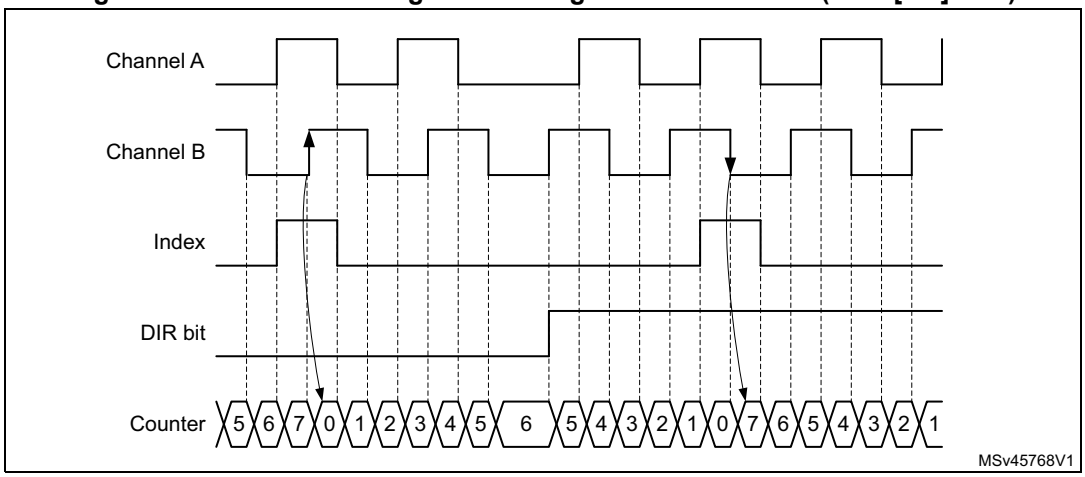


Figure 494 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 494. Counter reading with index ungated (IPOS[1:0] = 00)

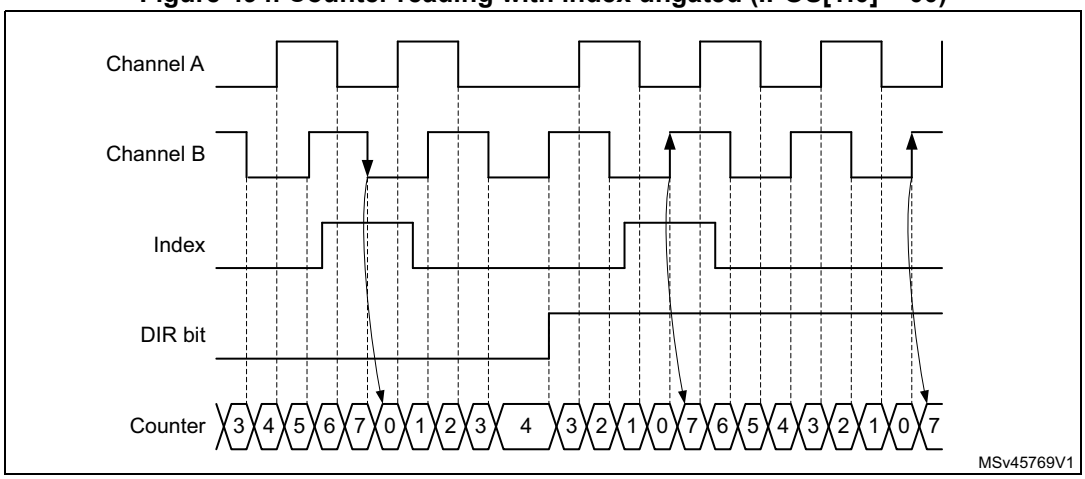


Figure 495 shows how the ‘gated on A & B’ mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 495. Counter reading with index gated on channel A and B

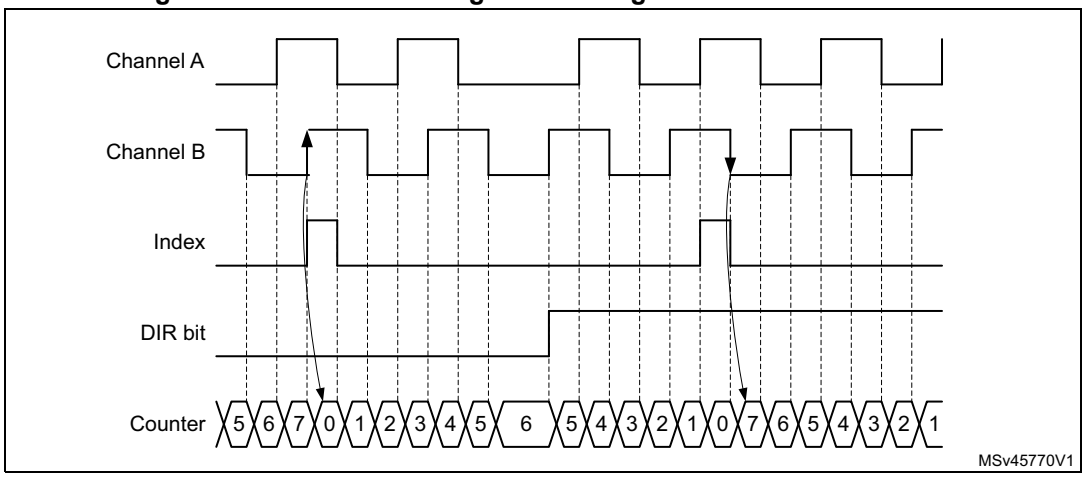


Figure 496 and *Figure 497* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 496. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

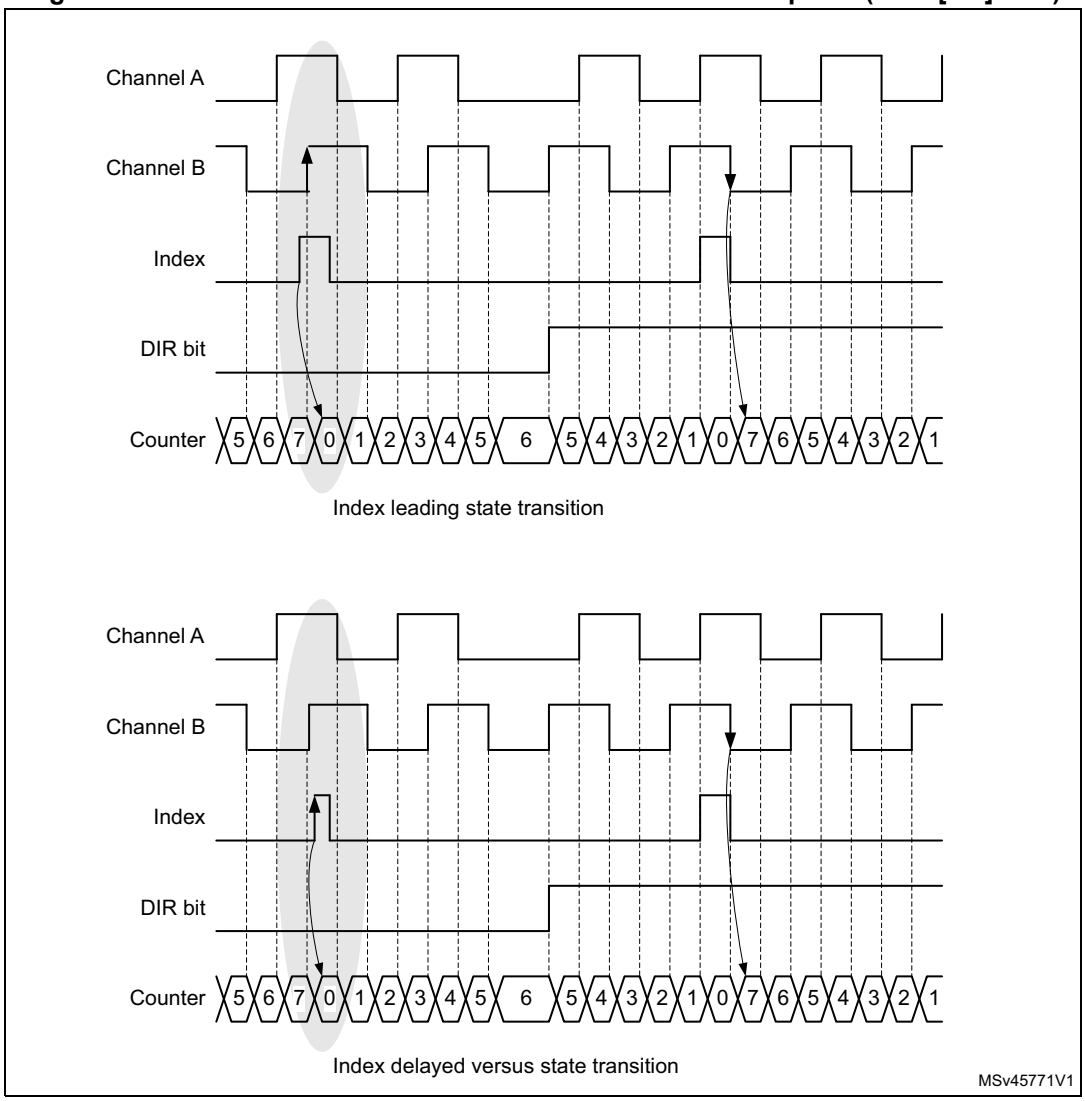


Figure 497. Counter reset Narrow index pulse (closer view, ARR = 0x07)

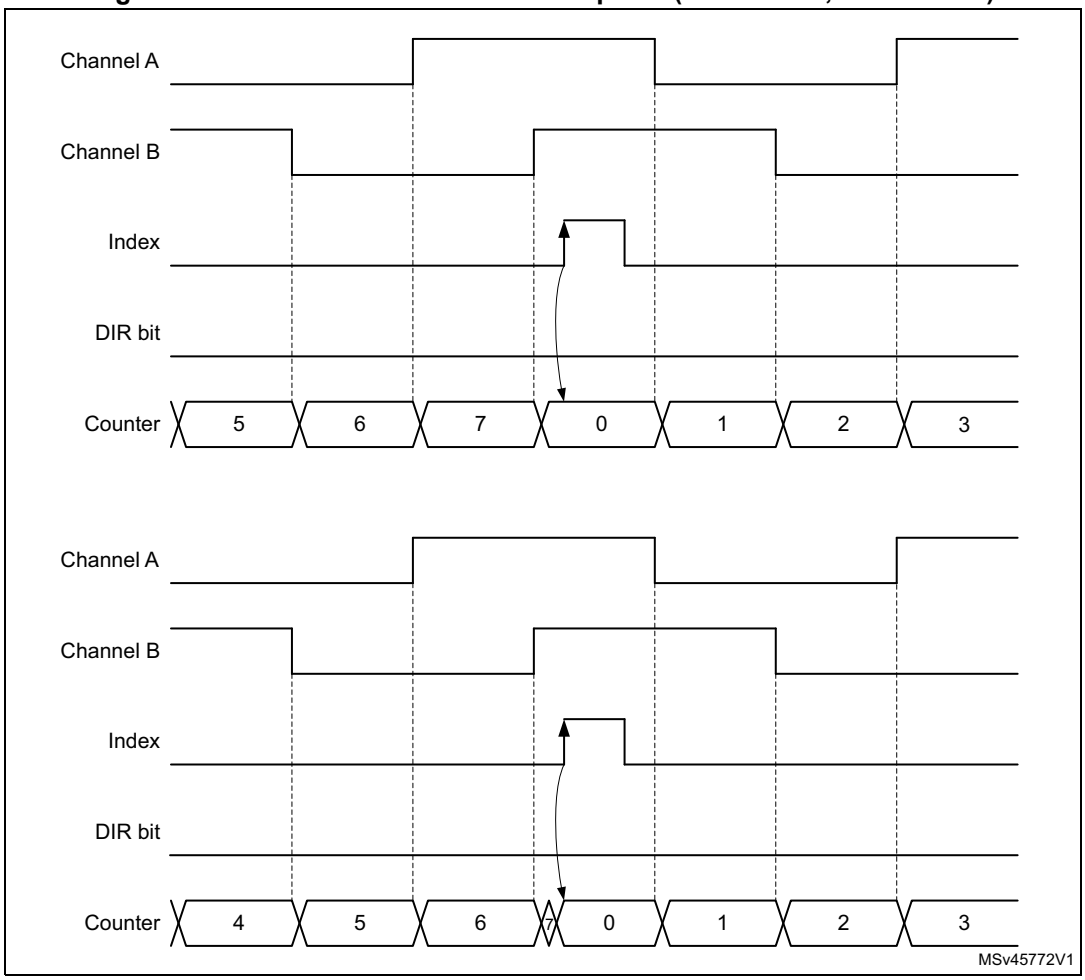
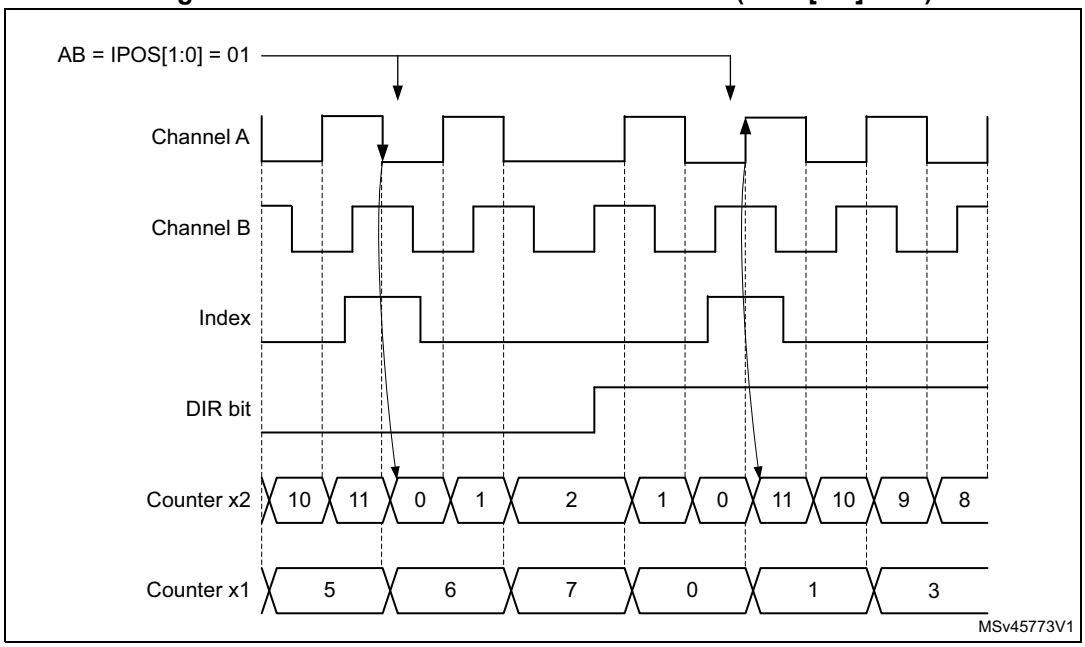


Figure 498 shows how the index is managed in x1 and x2 modes.

Figure 498. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)



Directional index sensitivity

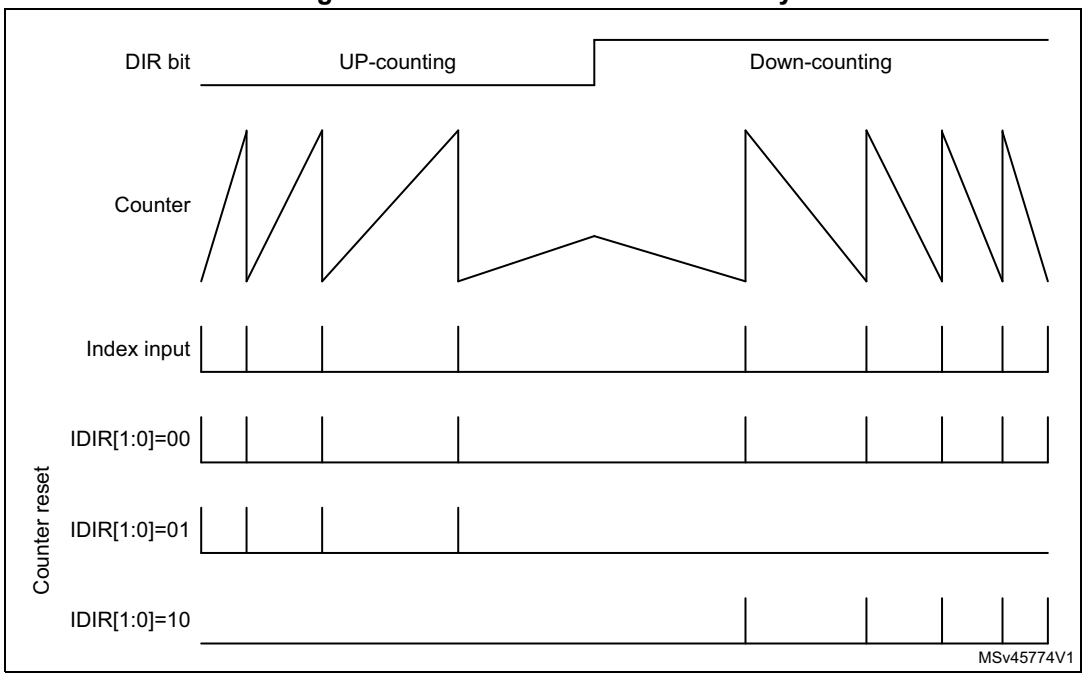
The IDIR[1:0] bit field in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 499 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

Note: The IDR[1:0] bit field must be written when IE bit is reset (index mode disabled).

Note: The directional index sensitivity is not supported in clock + direction mode. When SMS[3:0] = 1010 or 1011, the IDIR[1:0] must be set to 00.

Figure 499. Directional index sensitivity

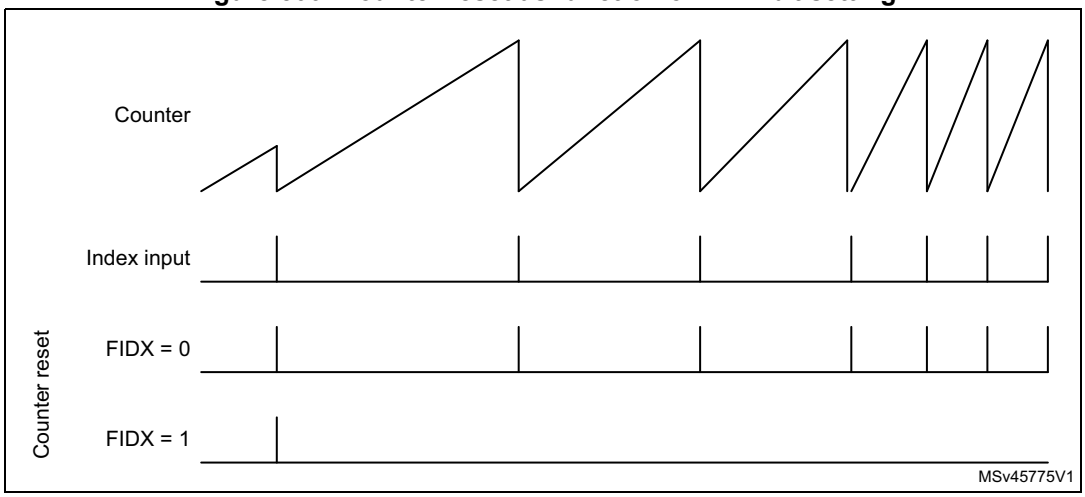


Special first index event management

The FIDX bit in the TIMx_ECR register allows the Index to be taken only once, as shown in *Figure 500*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be re-armed by writing the FIDX bit to 0 and setting it again to 1.

Note: When FIDX = 1, the index can be issued twice (IDXF flag set) if the direction changes at position 0 (index active).

Figure 500. Counter reset as function of FIDX bit setting



Index management in non-quadrature mode

Figure 501 and *Figure 502* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bit field is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as follows:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 501. Index behavior in clock + direction mode, IPOS[0] = 1

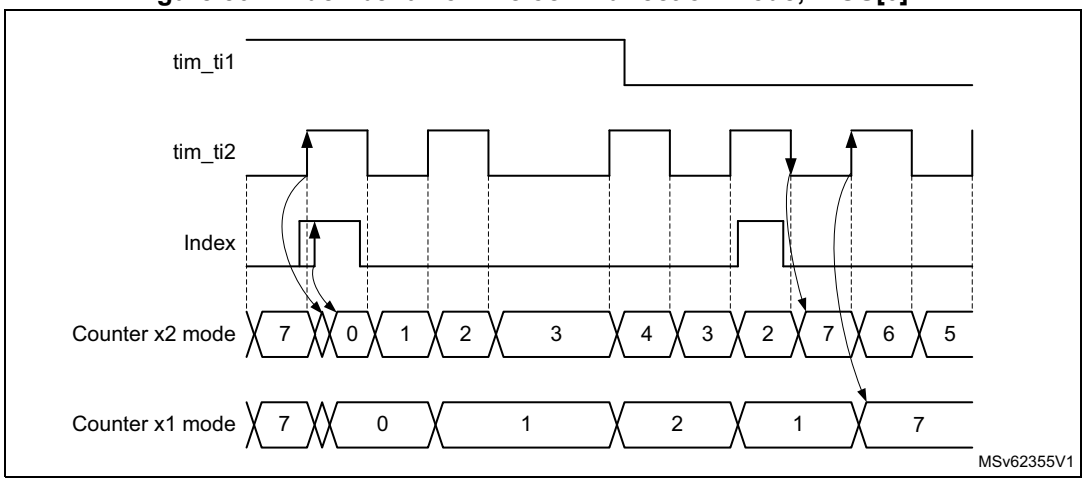
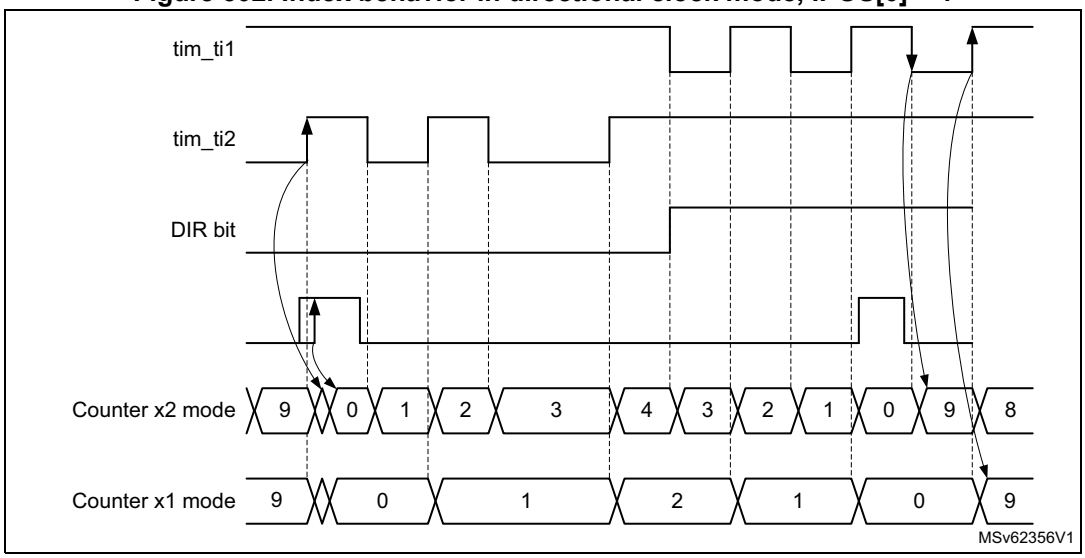


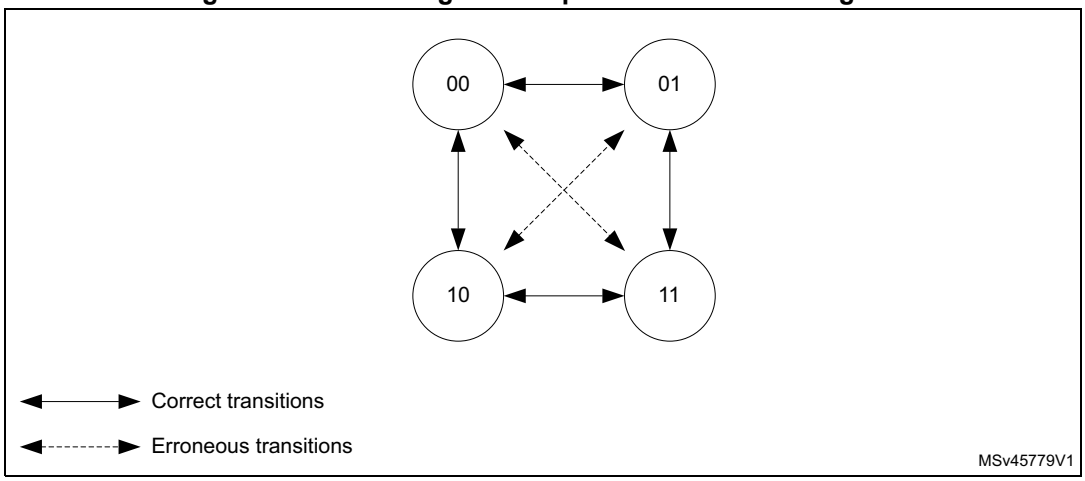
Figure 502. Index behavior in directional clock mode, IPOS[0] = 1



Encoder error management

For encoder configurations where 2 quadrature signals are available, it is possible to detect transition errors. The reading on the 2 inputs corresponds to a 2-bit gray code which can be represented as a state diagram, in *Figure 503*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 503. State diagram for quadrature encoded signals



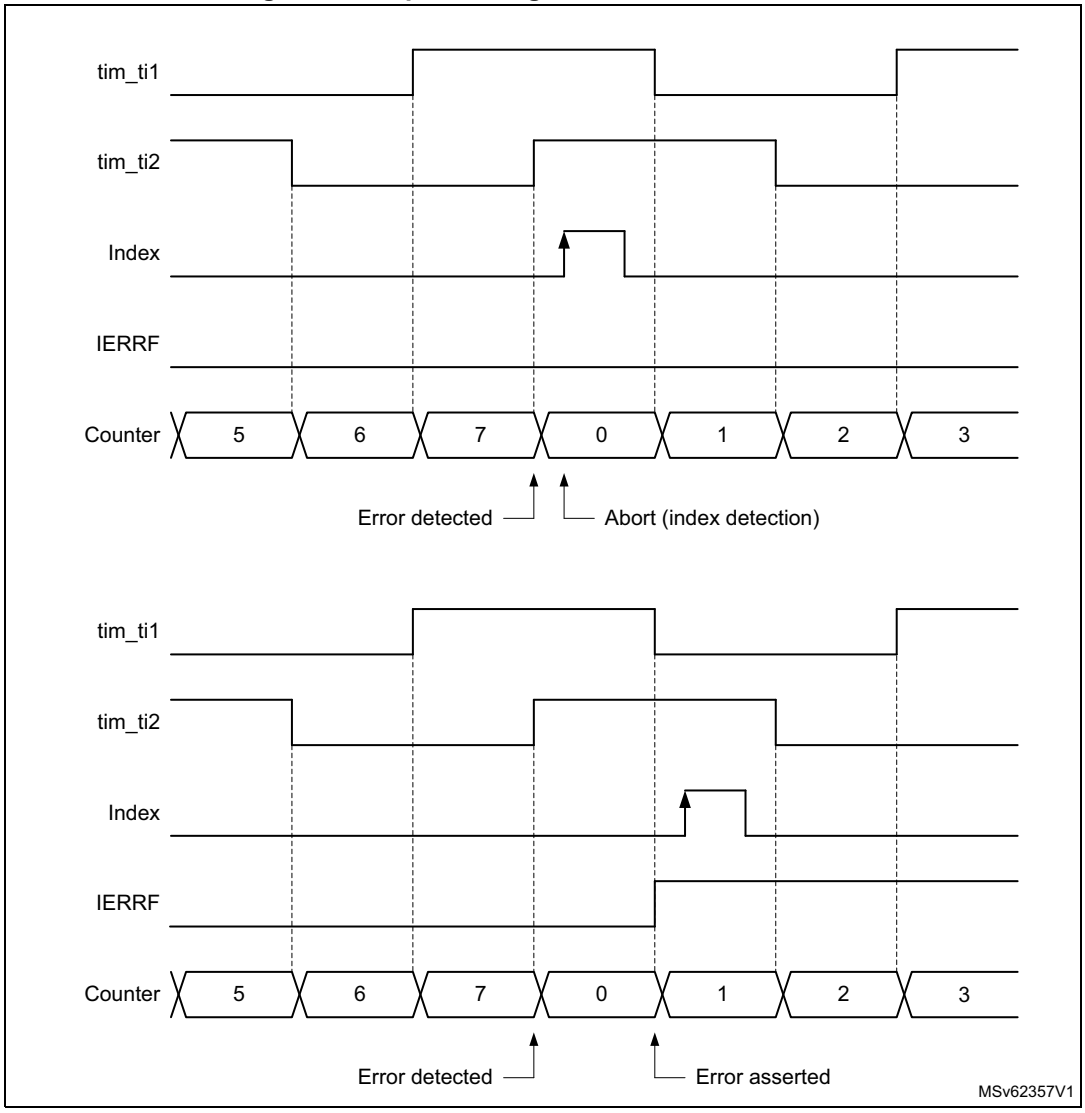
For encoder having an Index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The Index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an Index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

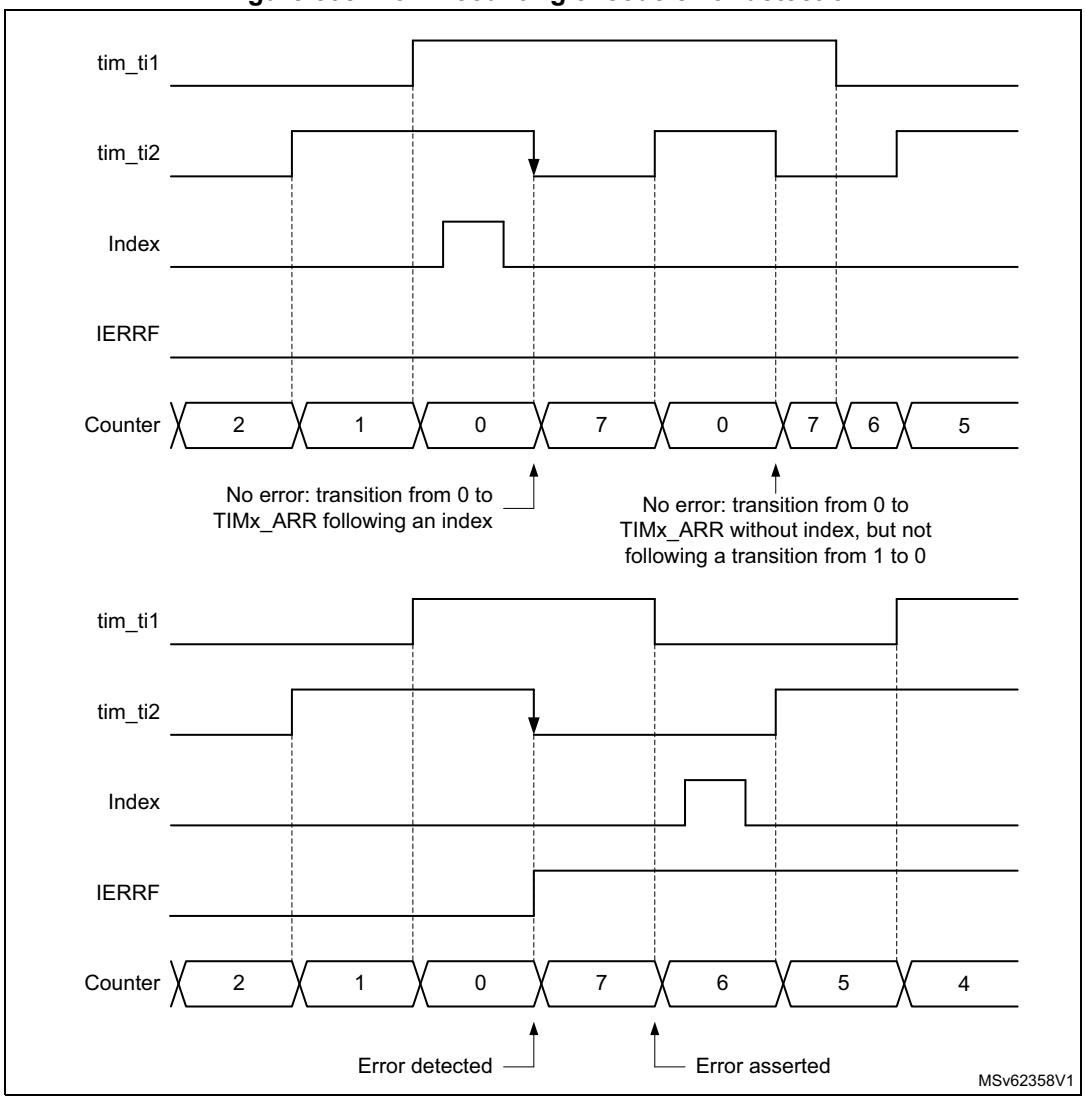
The error assertion is delayed to the transition 0 to 1 when in up-counting. This copes with narrow index pulses in gated A and B mode, as shown in *Figure 504*.

Figure 504. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown in *Figure 505*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 505. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder Interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the Index event sets the IDXF interrupt flag in the TIMx_SR status register. An Index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

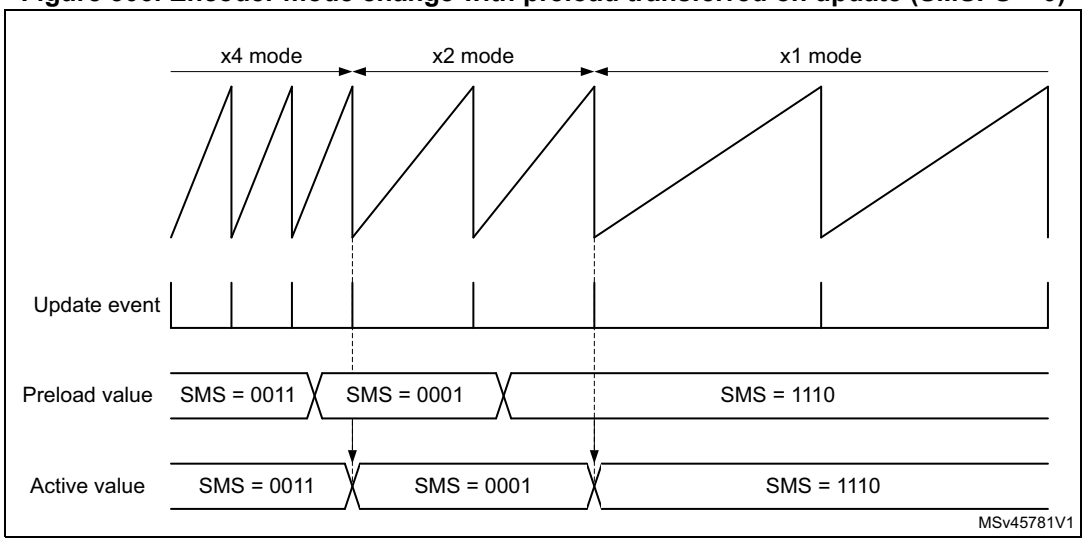
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the Update interrupt rate, by switching from x4 to x2 to x1 mode, as shown in *Figure 506*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting. This mode must be used only when index is disabled (bit IE = 0)
- SMSPS = 1: the transfer is triggered by the Index event

Figure 506. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bit field to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

35.3.26 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bit field to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

35.3.27 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

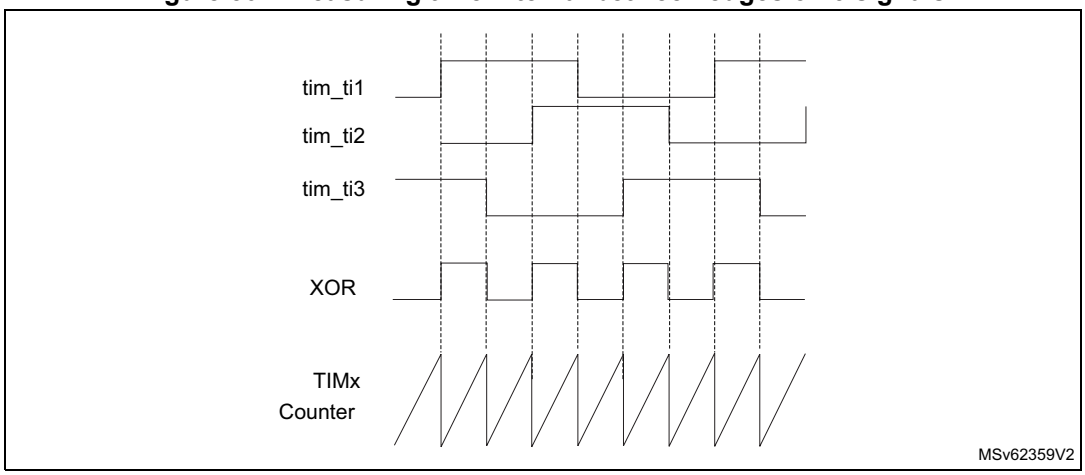
There is no latency between the UIF and UIFCPY flags assertion.

35.3.28 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 507*.

Figure 507. Measuring time interval between edges on 3 signals



35.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as "interfacing timer" in *Figure 508*. The "interfacing timer" captures the 3 timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the "interfacing timer", capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (refer to *Figure 451*). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

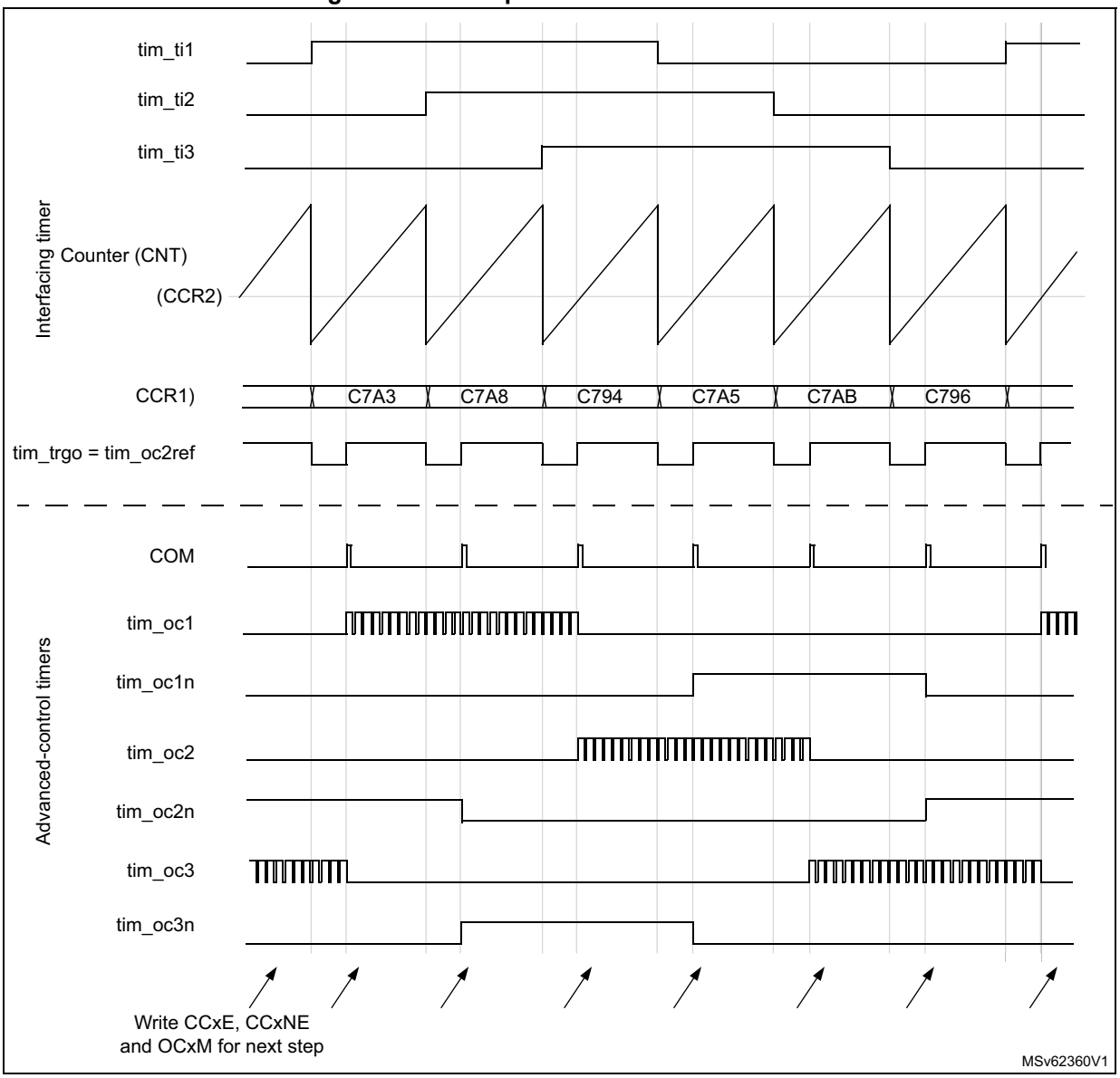
Example: one wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers:

- Configure 3 timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the tim_ti1 change). Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIMx_CCMR1 register to ‘01’. The digital filter can also be programmed if needed
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to ‘101’

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 508 describes this example.

Figure 508. Example of Hall sensor interface



35.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 36.4.23: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

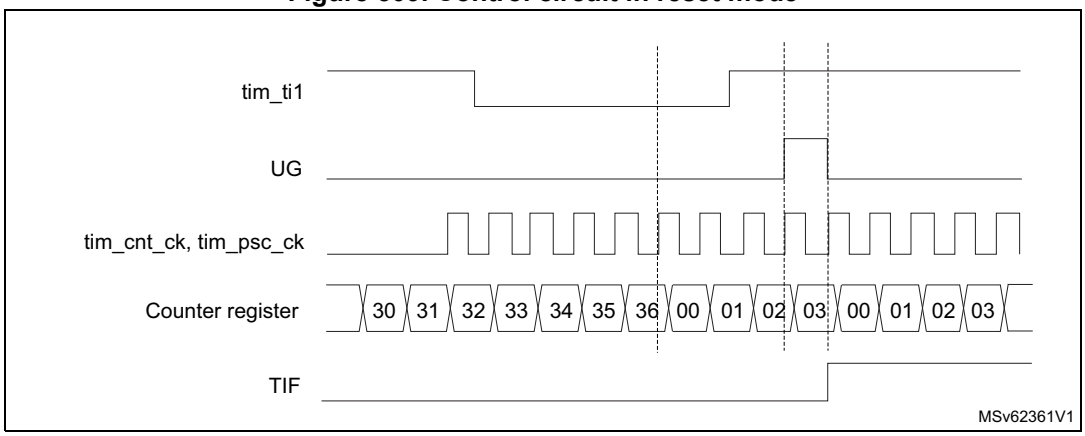
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 509. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

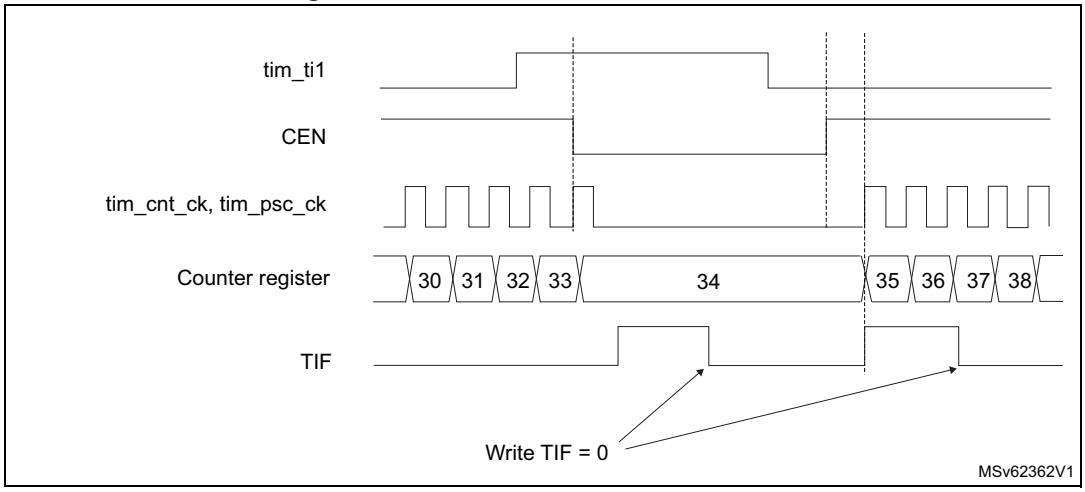
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 510. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

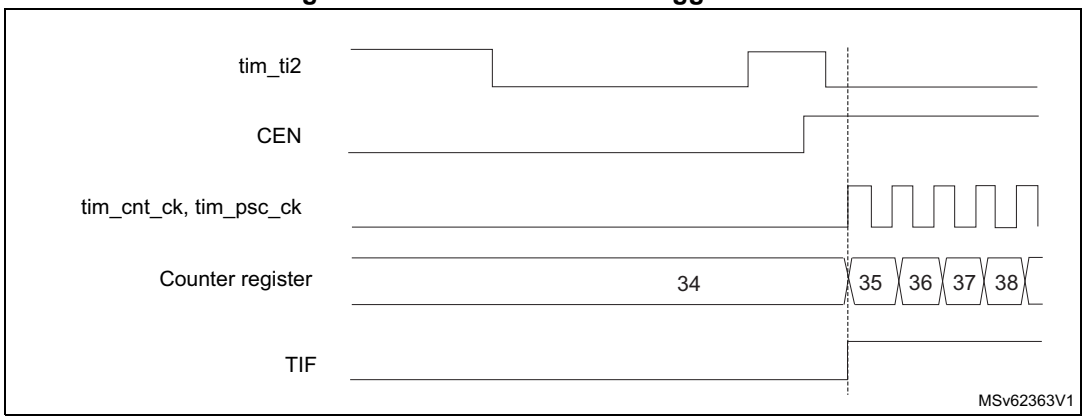
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 511. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode allows to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

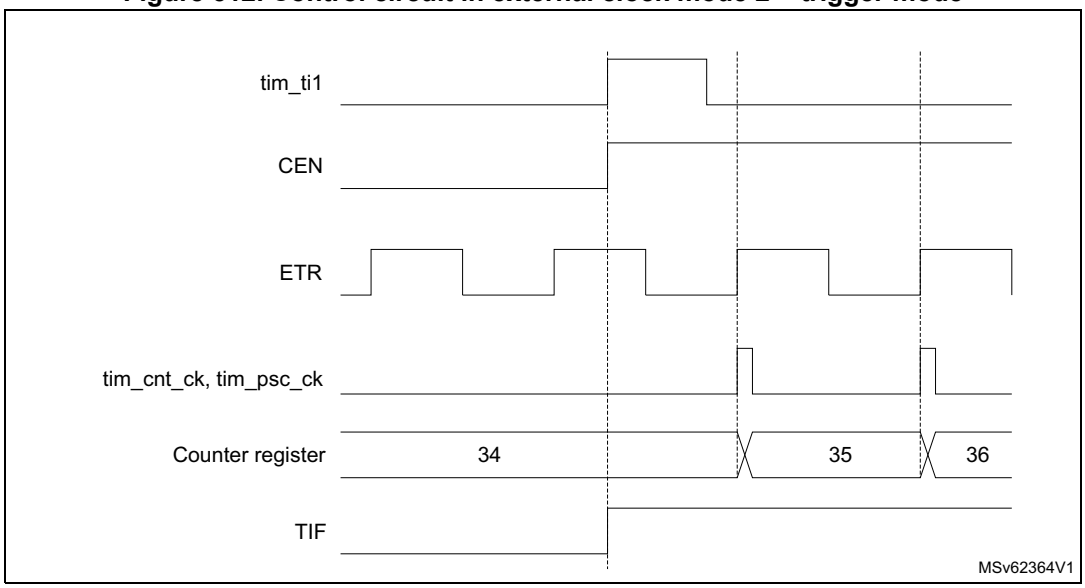
1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on tim_etr_in and ECE=1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 512. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.3.31 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the internal lines which are redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2 and MMS fields in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 466*.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

35.3.32 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), that is the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers
   - Number of data to transfer = 3 (refer to note below)
   - Circular mode disabled
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register)
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6.

Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

35.3.33 TIM1/TIM8 DMA requests

The TIM1/TIM8 can generate a DMA request, as shown in *Table 418*.

Table 418. DMA request

DMA acronym	DMA request	Enable control bit
TIM_UP	Update	UDE
TIM_CH1	Capture/compare 1	CC1DE
TIM_CH2	Capture/compare 2	CC2DE
TIM_CH3	Capture/compare 3	CC3DE
TIM_CH4	Capture/compare 4	CC4DE
TIM_COM	Commutation (COM)	COMDE
TIM_TRIG	Trigger	TDE

35.3.34 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stop, depending on TIMx_FRZ configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to *Chapter 10: Debug support (DBG)*.

35.4 TIM1/TIM8 low-power modes

Table 419. Effect of low-power modes on TIM1/TIM8

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.

35.5 TIM1/TIM8 interrupts

The TIM1/TIM8 can generate multiple interrupts, as shown in *Table 420*.

Table 420. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
TIM_UP	Update	UIF	UIE	Write 0 in UIF	Yes
TIM_CC	Capture/compare 1	CC1IF	CC1IE	Write 0 in CC1IF	Yes
	Capture/compare 2	CC2IF	CC2IE	Write 0 in CC2IF	Yes
	Capture/compare 3	CC3IF	CC3IE	Write 0 in CC3IF	Yes
	Capture/compare 4	CC4IF	CC4IE	Write 0 in CC4IF	Yes
TIM_TRG_COM	Commutation (COM)	COMIF	COMIE	Write 0 in COMIF	Yes
	Trigger	TIF	TIE	Write 0 in TIF	Yes
TIM_DIR_IDX	Index	IDXF	IDXIE	Write 0 in IDXF	Yes
	Direction	DIRF	DIRIE	Write 0 in DIRF	Yes
TIM_BRK	Break	BIF	BIE	Write 0 in BIF	Yes
	Break2	B2IF		Write 0 in B2IF	Yes
	System Break	SBIF		Write 0 in SBIF	Yes
TIM_IERR	Index Error	IERRF	IERRIE	Write 0 in IERRF	Yes
TIM_TER	Transition Error	TERRF	TERRIE	Write 0 in TERRF	Yes

35.6 TIM1/TIM8 registers

35.6.1 TIM1/TIM8 memory map

Table 421. TIM1/TIM8 register memory map

Offset	Register name
0x000	*TIMx control register 1 (TIMx_CR1)(x = 1, 8)*
0x004	*TIMx control register 2 (TIMx_CR2)(x = 1, 8)*
0x008	*TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)*
0x00C	*TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)*
0x010	*TIMx status register (TIMx_SR)(x = 1, 8)*
0x014	*TIMx event generation register (TIMx_EGR)(x = 1, 8)*
0x018	*TIMx capture/compare mode register 1 [alternate](TIMx_CCMR1)(x = 1, 8)*
0x018	*TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)*
0x01C	*TIMx capture/compare mode register 2 [alternate](TIMx_CCMR2)(x = 1, 8)*
0x01C	*TIMx capture/compare mode register 2 [alternate](TIMx_CCMR2)(x = 1, 8)*
0x020	*TIMx capture/compare enable register(TIMx_CCER)(x = 1, 8)*
0x024	*TIMx counter (TIMx_CNT)(x = 1, 8)*
0x028	*TIMx prescaler (TIMx_PSC)(x = 1, 8)*
0x02C	*TIMx auto-reload register (TIMx_ARR)(x = 1, 8)*
0x030	*TIMx repetition counter register (TIMx_RCR)(x = 1, 8)*
0x034	*TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)*
0x038	*TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)*
0x03C	*TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)*
0x040	*TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)*
0x044	*TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)*
0x048	*TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)*
0x04C	*TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)*
0x050	*TIMx capture/compare mode register 3 (TIMx_CCMR3)(x = 1, 8)*
0x054	*TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8)*
0x058	*TIMx timer encoder control register (TIMx_ECR)(x = 1, 8)*
0x05C	*TIMx timer input selection register (TIMx_TISEL)(x = 1, 8)*
0x060	*TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8)*
0x064	*TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8)*
0x3DC	*TIMx DMA control register (TIMx_DCR)(x = 1, 8)*
0x3E0	*TIMx DMA address for full transfer (TIMx_DMAR) (x = 1, 8)*

35.6.2 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit field indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS}=t_{tim_ker_ck}$
01: $t_{DTS}=2*t_{tim_ker_ck}$
10: $t_{DTS}=4*t_{tim_ker_ck}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

35.6.3 TIMx control register 2 (TIMx_CR2)(x = 1, 8)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
						rw		rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

1000: **Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

1001: **Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

1010: **Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

1011: **Compare Pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

1100: **Compare Pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1101: **Compare Pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

1110: **Compare Pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1111: **Compare Pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)
0: tim_oc1n=0 after a dead-time when MOE=0
1: tim_oc1n=1 after a dead-time when MOE=0
Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)
0: tim_oc1=0 (after a dead-time) when MOE=0
1: tim_oc1=1 (after a dead-time) when MOE=0
Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15..0] multiplexer output is connected to tim_ti1 input
1: tim_ti1_in[15..0], tim_ti2_in[15..0] and tim_ti3_in[15..0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection
These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:
0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.
0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.
0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).
0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)
0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)
0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)
0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)
1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.
Other codes reserved
Note: The clock of the slave timer or ADC must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.

35.6.4 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source
This bit selects whether the events that trigger the SMS[3:0] bit field transfer from preload to active
0: The transfer is triggered by the Timer's Update event
1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable
This bit selects whether the SMS[3:0] bit field is preloaded
0: SMS[3:0] bit field is not preloaded
1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3
Refer to TS[2:0] description - bits 6:4

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity
This bit selects whether tim_etr_in or tim_$\overline{\text{etr_in}}$ is used for trigger operations
0: tim_etr_in is non-inverted, active at high level or rising edge.
1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

*Note: **1:** Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS=111 and TS=00111).*

***2:** It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).*

***3:** If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF

01: tim_etr_in frequency divided by 2

10: tim_etr_in frequency divided by 4

11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit field then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2

0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4

0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bit field is combined with TS[4:3] bits.

This bit field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External Trigger input (tim_etrf)
01000: Internal Trigger 0 (tim_itr4)
01001: Internal Trigger 1 (tim_itr5)
01010: Internal Trigger 1 (tim_itr6)
01011: Internal Trigger 1 (tim_itr7)
01100: Internal Trigger 1 (tim_itr8)
01101: Internal Trigger 1 (tim_itr9)
01110: Internal Trigger 1 (tim_itr10)
others: Reserved

Refer to the Device Configuration chapter for more details on tim_itrx meaning for each Timer.

Note: These bits must be changed only when they are not used (for example, when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Bits 16, 2:0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.6.5 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

35.6.6 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF
								rc_w0	rc_w0	rc_w0	rc_w0			rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to ‘0’.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to ‘0’.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to ‘0’.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to ‘0’.
0: No index event occurred.
1: An index event has occurred

Bits 19:14 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on tim_ic1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

–At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

–When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

–When CNT is reinitialized by a trigger event (refer to *Section 35.6.4: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

35.6.7 TIMx event generation register (TIMx_EGR)(x = 1, 8)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

35.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode:

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2

0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4

0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit field defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 Selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

35.6.9 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref=‘0’) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref=’1’).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channel becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channel becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channel becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channel becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: These bits cannot be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits cannot be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

35.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bit field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bit field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

35.6.11 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] bit description

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bit field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT=TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In upcounting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref='0') as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref='1').

0111: PWM mode 2 - In upcounting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In downcounting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channel becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channel becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channel becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channel becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bit fields programming in TIMxECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits cannot be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

35.6.12 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

Bit 13 **CC4P**: Capture/compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high.
1: tim_oc1n active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/compare 1 output polarity

CC1 channel configured as output:

0: tim_oc1 active high

1: tim_oc1 active low

CC1 channel configured as input: CC1NP/CC1P bits select the active polarity of tim_ti1fp1 and tim_ti2fp1 for trigger or capture operations.

00: non-inverted/rising edge. The circuit is sensitive to tim_tixfp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode or encoder mode).

01: inverted/falling edge. The circuit is sensitive to tim_tixfp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is inverted (trigger operation in gated mode or encoder mode).

10: reserved, do not use this configuration.

11: non-inverted/both edges/ The circuit is sensitive to both tim_tixfp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

CC1 channel configured as output:

0: Off - tim_oc1 is not active. tim_oc1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - tim_oc1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled.

1: Capture enabled.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 422. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0, tim_ocxn = 0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref x or CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP (if tim_brk or tim_brk2 is triggered). Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration). *Note: tim_brk2 can only be used if OSSI = OSSR = 1.*	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

35.6.13 TIMx counter (TIMx_CNT)(x = 1, 8)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

35.6.14 TIMx prescaler (TIMx_PSC)(x = 1, 8)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

35.6.15 TIMx auto-reload register (TIMx_ARR)(x = 1, 8)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to *Section 35.3.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bit field contains the dithered part.

35.6.16 TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x030

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: Repetition counter reload value

This bit field defines the update rate of the compare registers (that are periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

35.6.17 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bit field contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

35.6.18 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bit field contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

35.6.19 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bit field contains the dithered part.

If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

35.6.20 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bit field contains the dithered part.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

35.6.21 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional

Refer to BKBID description

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode

1: Break input tim_brk in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable
This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 471: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled
Note: The BRKIN2 must only be used with OSSR = OSSI = 1.
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter
This bit field defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
0000: No filter, tim_brk2 acts asynchronously
0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bit field defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).
See OC/OCN enable description for more details (*Section 35.6.12: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 471: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

Refer to OC/OCN enable description for more details (*Section 35.6.12: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit cannot be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.

See OC/OCN enable description for more details (*Section 35.6.12: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit cannot be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BK2BID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit field defines the duration of the dead-time inserted between the complementary outputs. DT corresponds to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.
DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us steps

Note: This bit field cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

35.6.22 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw										rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3

Distortion on channel 3 output:
0: No effect of tim_oc5ref on tim_oc3refc
1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2

Distortion on channel 2 output:
0: No effect of tim_oc5ref on tim_oc2refc
1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1

Distortion on channel 1 output:
0: No effect of oc5ref on oc1refc
1: oc1refc is the logical AND of oc1ref and oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: Capture/compare 5 value

CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc5 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bit field contains the dithered part.

35.6.23 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR6[19:0]**: Capture/compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc6 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bit field contains the dithered part.

35.6.24 TIMx capture/compare mode register 3 (TIMx_CCMR3)(x = 1, 8)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

Bits 24, 14:12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

Bits 16, 6:4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, must be kept at reset value.

35.6.25 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.

DTGF[7:5]=10x => DTF=(64+DTGF[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.

DTGF[7:5]=110 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.

DTGF[7:5]=111 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

Note: This bit field cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

35.6.26 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	Res.	Res.	IDIR[1:0]		IE
								rw	rw	rw			rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bit field sets the clock prescaler for the pulse generator, as follows:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bit field defines the pulse duration, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicate on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Note: The IDR[1:0] bit field must be written when IE bit is reset (index disabled).

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

35.6.27 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[0..15] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[0..15] input
0000: tim_ti3_in0: TIMx_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[0..15] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[0..15] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

35.6.28 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation.
Note: These bits cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input polarity is not inverted (active low if BKP=0, active high if BKP=1)
1: tim_brk_cmp4 input polarity is inverted (active high if BKP=0, active low if BKP=1)
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity
This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp3 input is not inverted (active low if BKP=0, active high if BKP=1)
1: tim_brk_cmp3 input is inverted (active high if BKP=0, active low if BKP=1)
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity
This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp2 input is not inverted (active low if BKP=0, active high if BKP=1)
1: tim_brk_cmp2 input is inverted (active high if BKP=0, active low if BKP=1)
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity
This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp1 input is not inverted (active low if BKP=0, active high if BKP=1)
1: tim_brk_cmp1 input is inverted (active high if BKP=0, active low if BKP=1)
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity
This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: TIMx_BKIN input is not inverted (active low if BKP=0, active high if BKP=1)
1: TIMx_BKIN input is inverted (active high if BKP=0, active low if BKP=1)
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 35.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

35.6.29 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8)

Address offset: 0x064

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

...

111: tim_ocref_clr7

Refer to *Section 35.3.2: TIM1/TIM8 pins and internal signals* for product specific information.

Note: These bits cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input is not inverted (active low if BK2P=0, active high if BK2P=1)

1: tim_brk2_cmp4 input is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input is not inverted (active low if BK2P=0, active high if BK2P=1)

1: tim_brk2_cmp3 input is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input is not inverted (active low if BK2P=0, active high if BK2P=1)

1: tim_brk2_cmp2 input is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input is not inverted (active low if BK2P=0, active high if BK2P=1)

1: tim_brk2_cmp1 input is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: TIMx_BKIN2 input polarity

This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIMx_BKIN2 input is not inverted (active low if BK2P=0, active high if BK2P=1)

1: TIMx_BKIN2 input is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: TIMx_BKIN2 input enable

This bit enables the TIMx_BKIN2 alternate function input for the timer's tim_brk2 input. TIMx_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIMx_BKIN2 input disabled

1: TIMx_BKIN2 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 35.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

35.6.30 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), that is the number of transfers. Transfers can be in half-words or in bytes (refer to example below).

00000: 1 transfer

00001: 2 transfers

00010: 3 transfers

...

11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer must be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access is done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

...

35.6.31 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

36 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

36.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 36.4.23: Timer synchronization*.

36.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

36.3 TIM2/TIM3/TIM4/TIM5 implementation

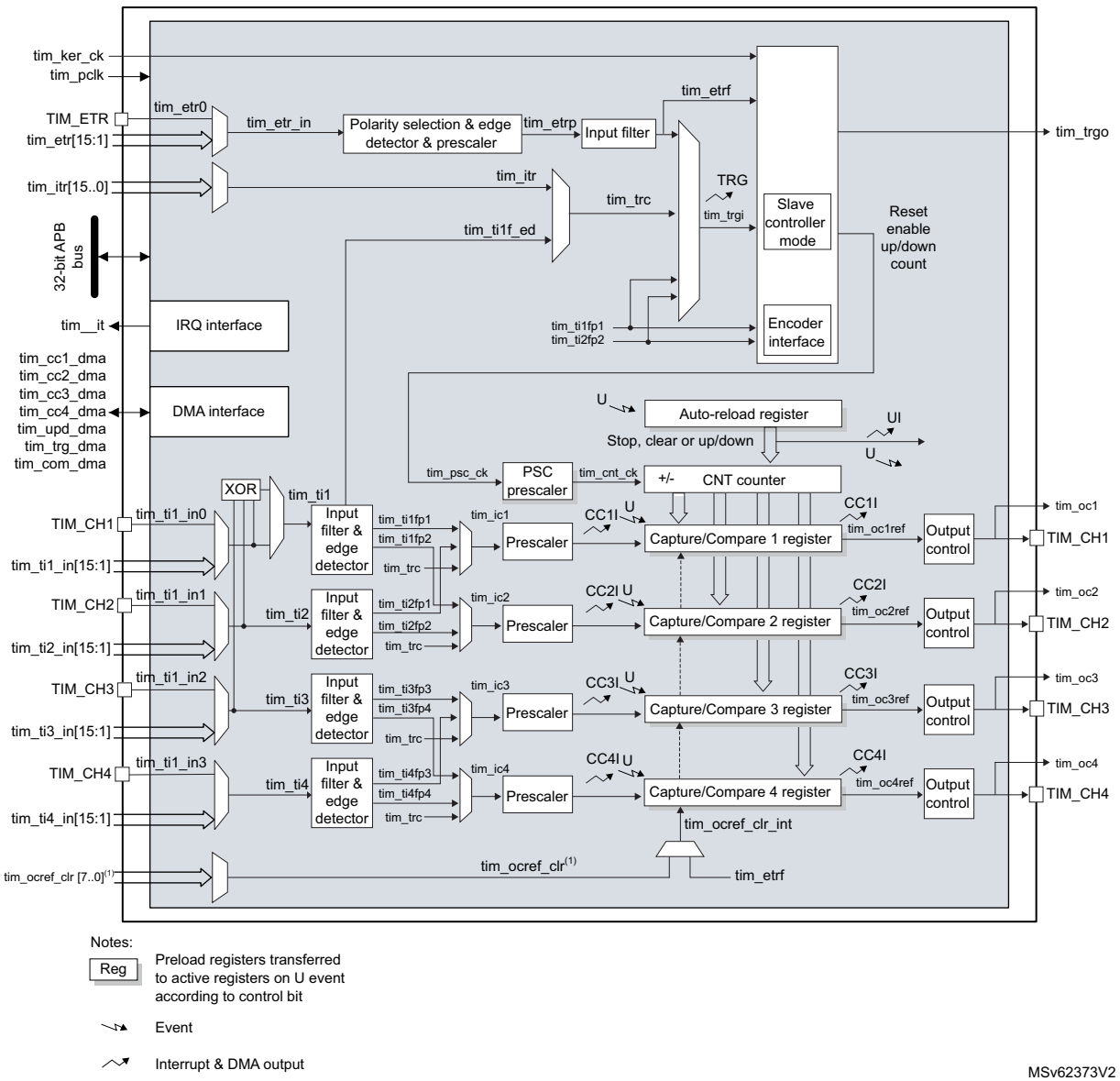
Table 423. SR5E1x general purpose timers

Timer instance	TIM2	TIM3	TIM4	TIM5
Resolution	32-bit	16-bit	16-bit	32-bit
OCREF clear selection	Yes	Yes	No	No
Sources	tim_etrf tim_ocref_clr[7:0]	tim_etrf tim_ocref_clr[7:0]	tim_etrf -	tim_etrf -

36.4 TIM2/TIM3/TIM4/TIM5 functional description

36.4.1 Block diagram

Figure 513. General-purpose timer block diagram



1. This feature is not available on all timers, refer to the *Section 36.3: TIM2/TIM3/TIM4/TIM5 implementation*.

36.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 424 and *Table 425* in this section summarize the TIM inputs and outputs.

Table 424. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 425. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.

Table 425. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 8 (maximum value).
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trg_dma	Output	Timer trigger dma request.
tim_com_dma	Output	Timer commutation dma request.

The sources connected to the tim_ti[1..4] input multiplexers are listed in *Section 5.5.4.1: Interconnects to the tim_ti[1..4] input multiplexers* from Device configuration chapter.

The internal sources connected to the tim_itr input multiplexer and the tim_etr input multiplexer are listed in *Section 5.5.4.2: Interconnects to the trigger input multiplexers* from Device configuration chapter.

The internal sources connected to the tim_ocref_clr input multiplexer are listed in *Section 5.5.4.3: Interconnects to the ocref_clr input multiplexer* from Device configuration chapter.

Table 426. Interconnect to the tim_ocref_clr input multiplexer

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr0	comp1_out	comp1_out	Reserved	Reserved
tim_ocref_clr1	comp2_out	comp2_out		
tim_ocref_clr2	comp3_out	comp3_out		
tim_ocref_clr3	comp4_out	comp4_out		
tim_ocref_clr4	comp5_out	comp5_out		
tim_ocref_clr5	comp6_out	comp6_out		
tim_ocref_clr6	comp7_out	comp7_out		
tim_ocref_clr7	Reserved			

36.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC):
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The contents of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 514 and *Figure 515* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 514. Counter timing diagram with prescaler division change from 1 to 2

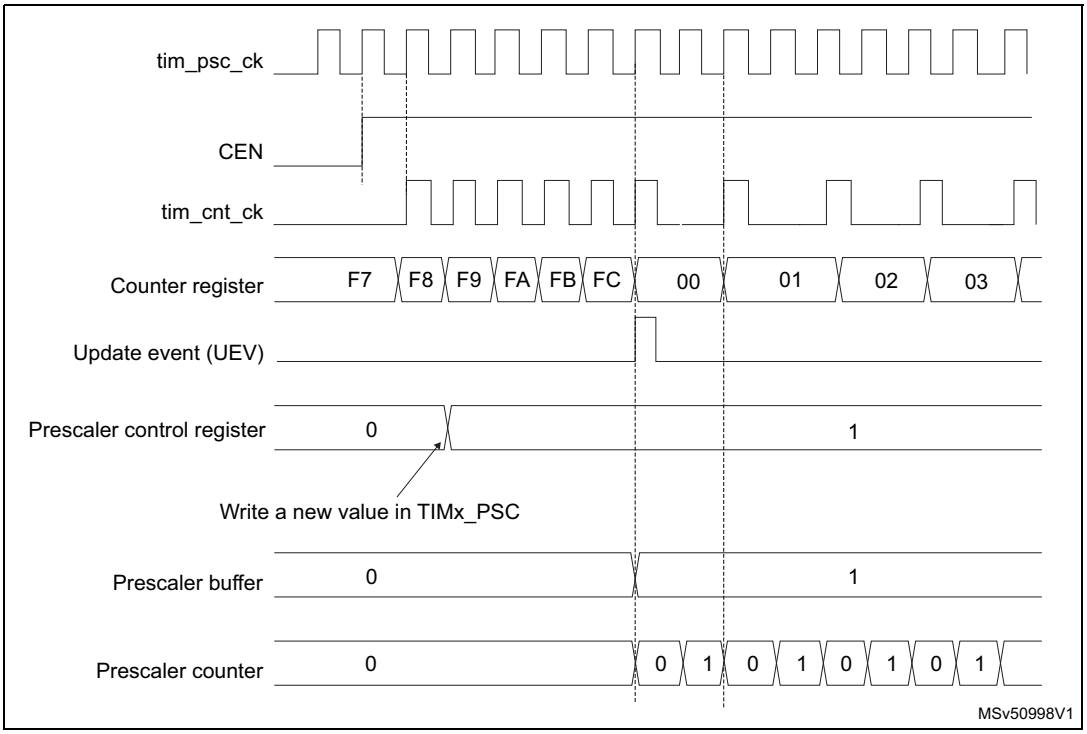
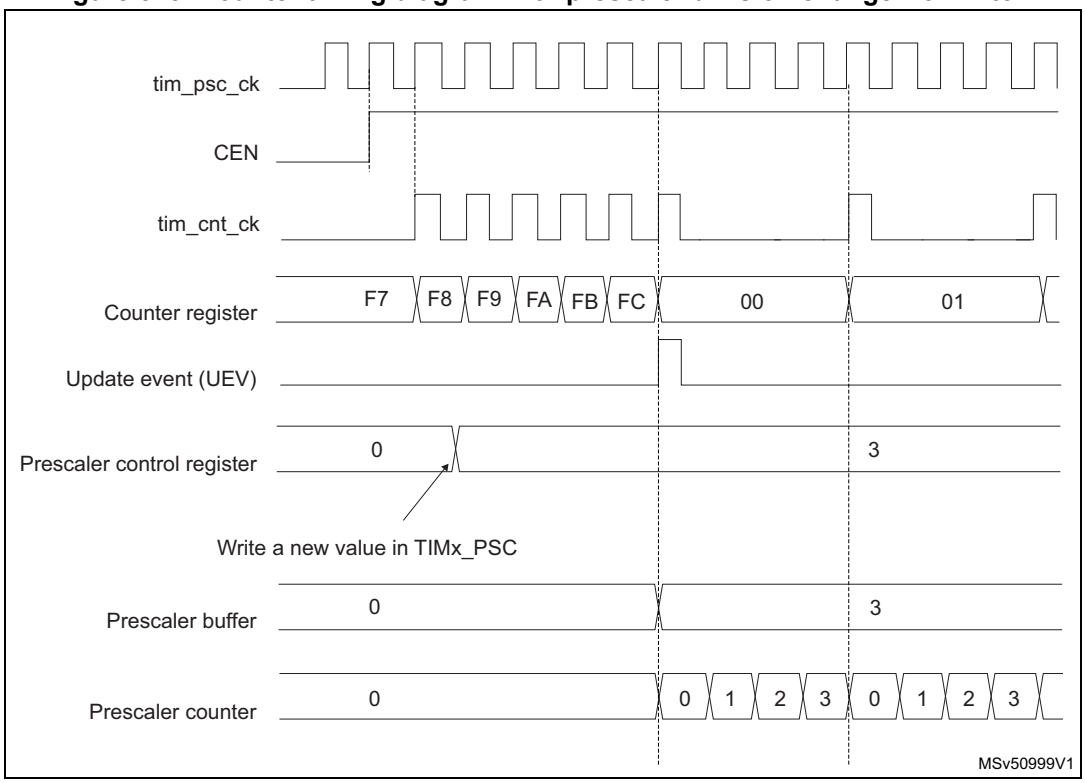


Figure 515. Counter timing diagram with prescaler division change from 1 to 4



36.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 516. Counter timing diagram, internal clock divided by 1

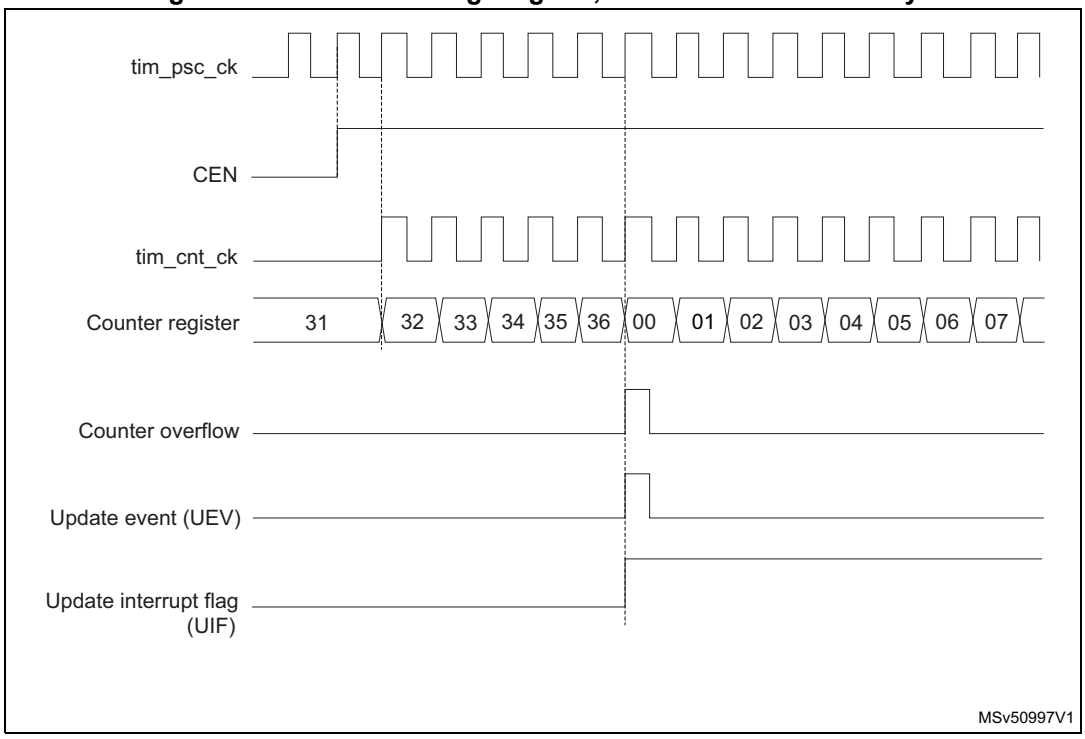


Figure 517. Counter timing diagram, internal clock divided by 2

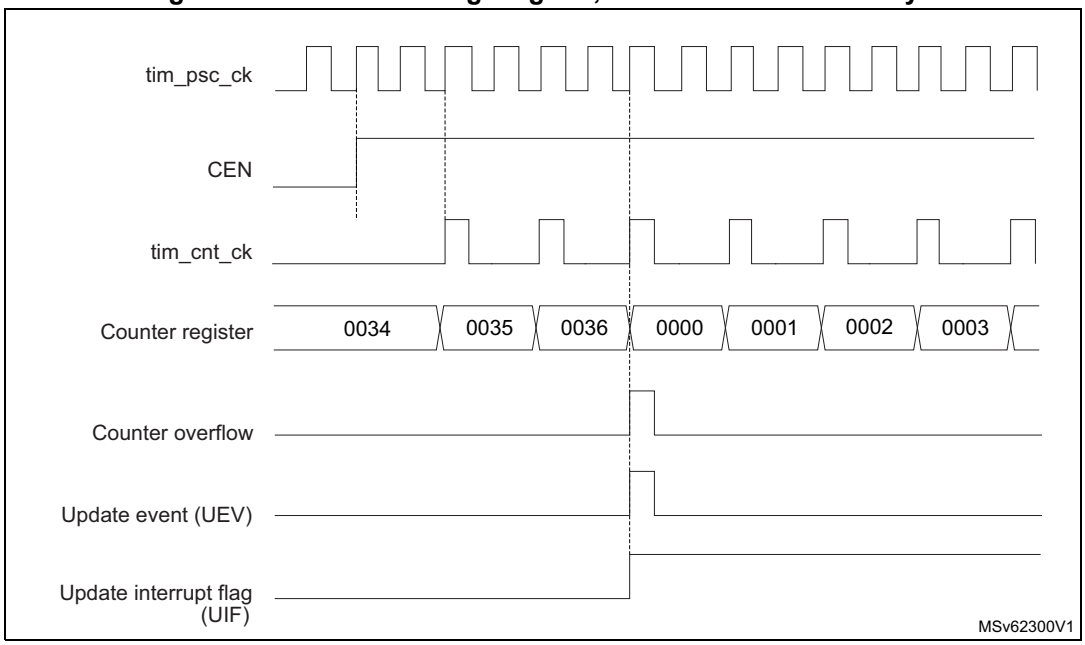


Figure 518. Counter timing diagram, internal clock divided by 4

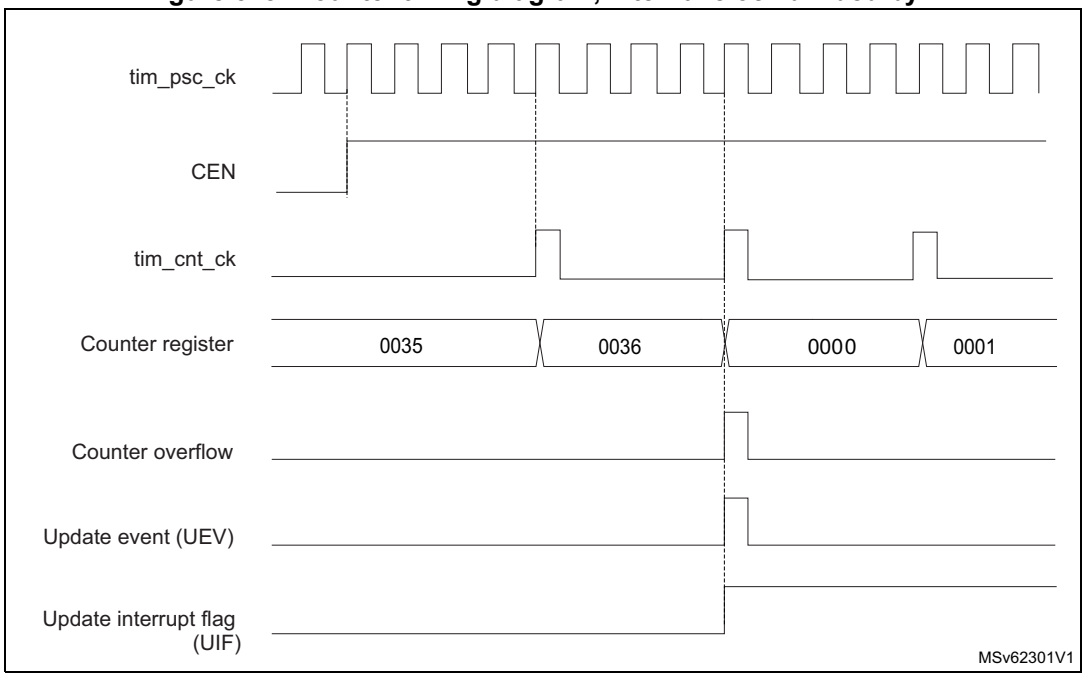


Figure 519. Counter timing diagram, internal clock divided by N

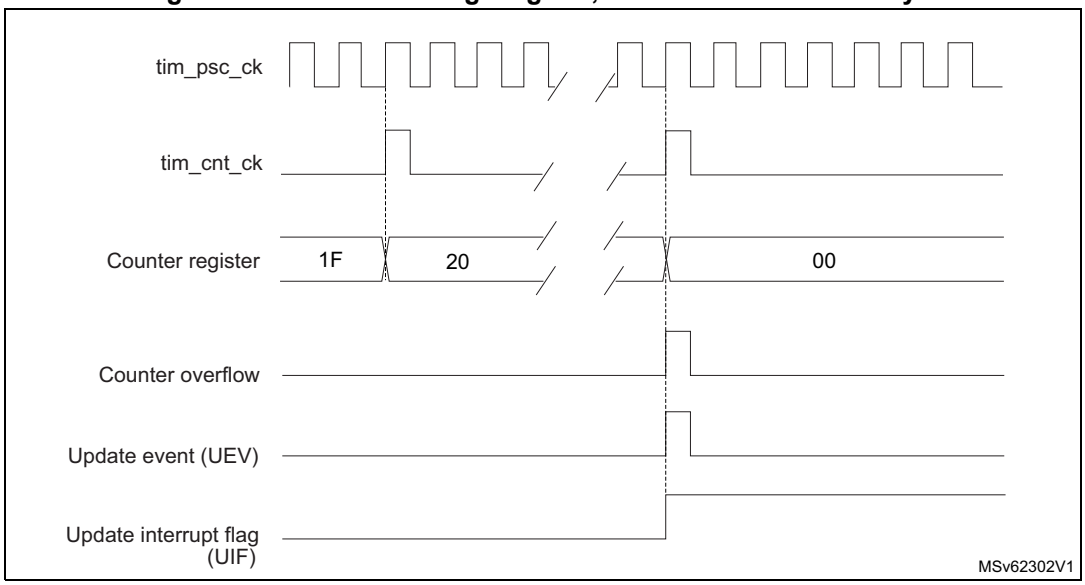


Figure 520. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

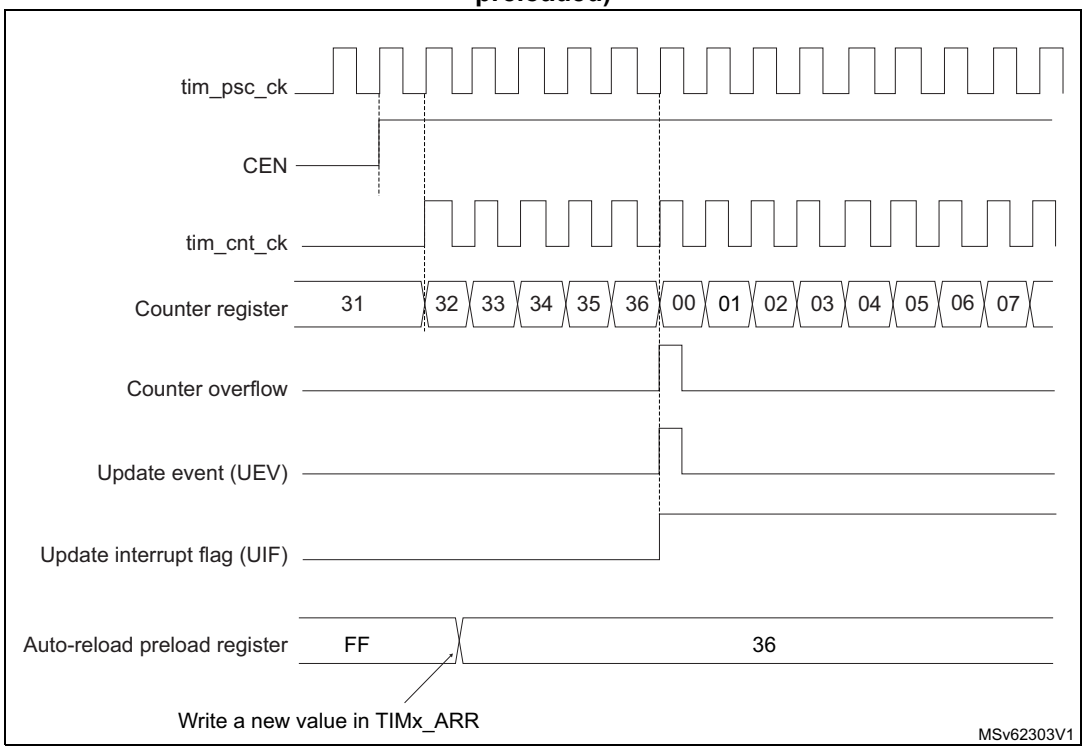
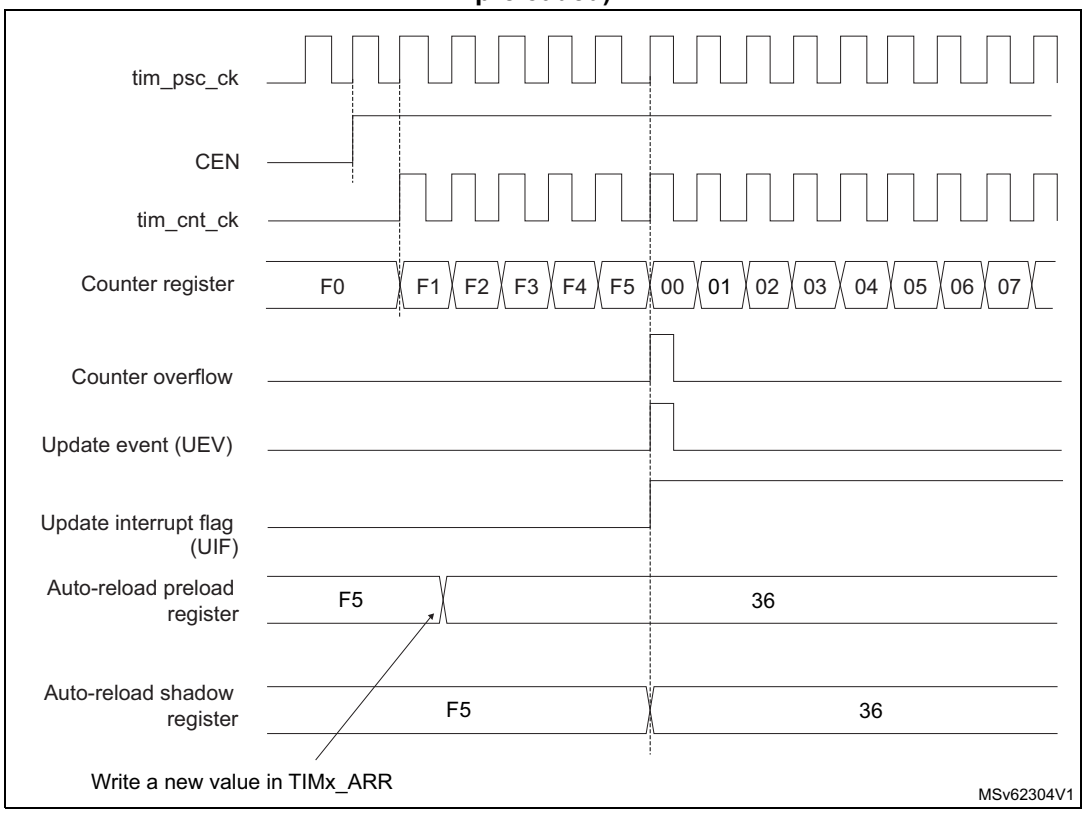


Figure 521. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generated at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 522. Counter timing diagram, internal clock divided by 1

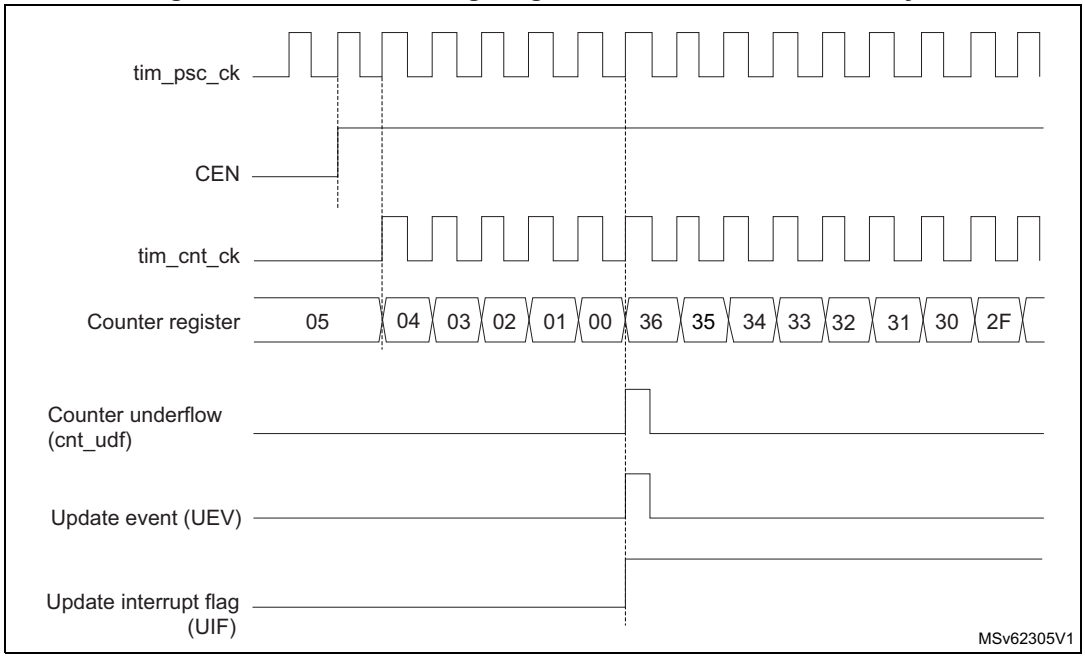


Figure 523. Counter timing diagram, internal clock divided by 2

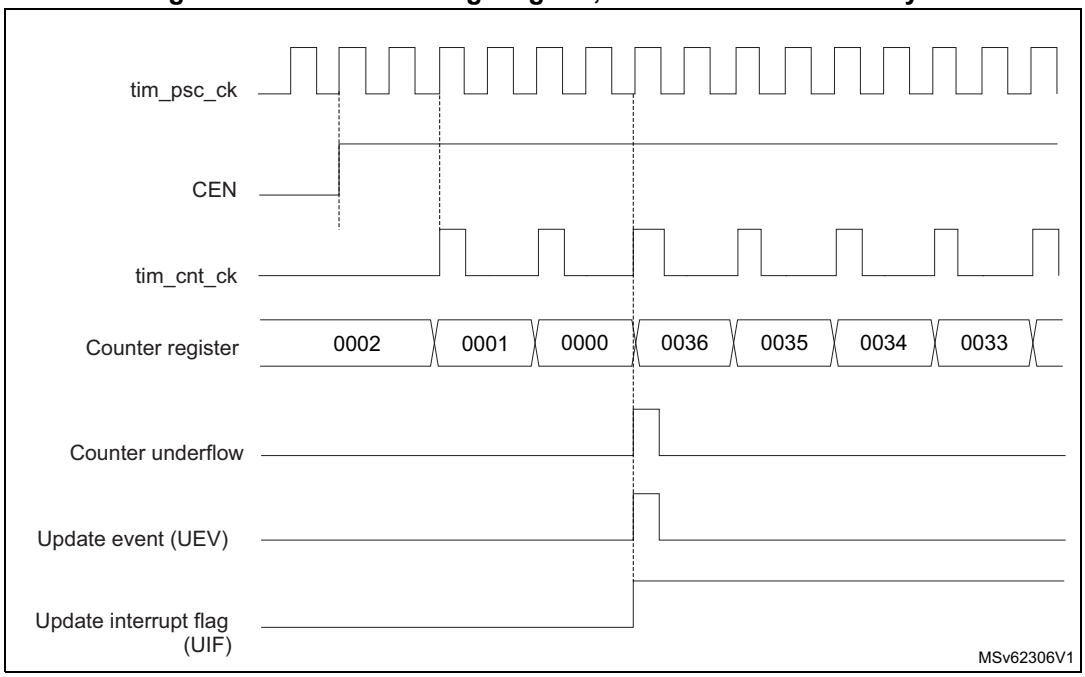


Figure 524. Counter timing diagram, internal clock divided by 4

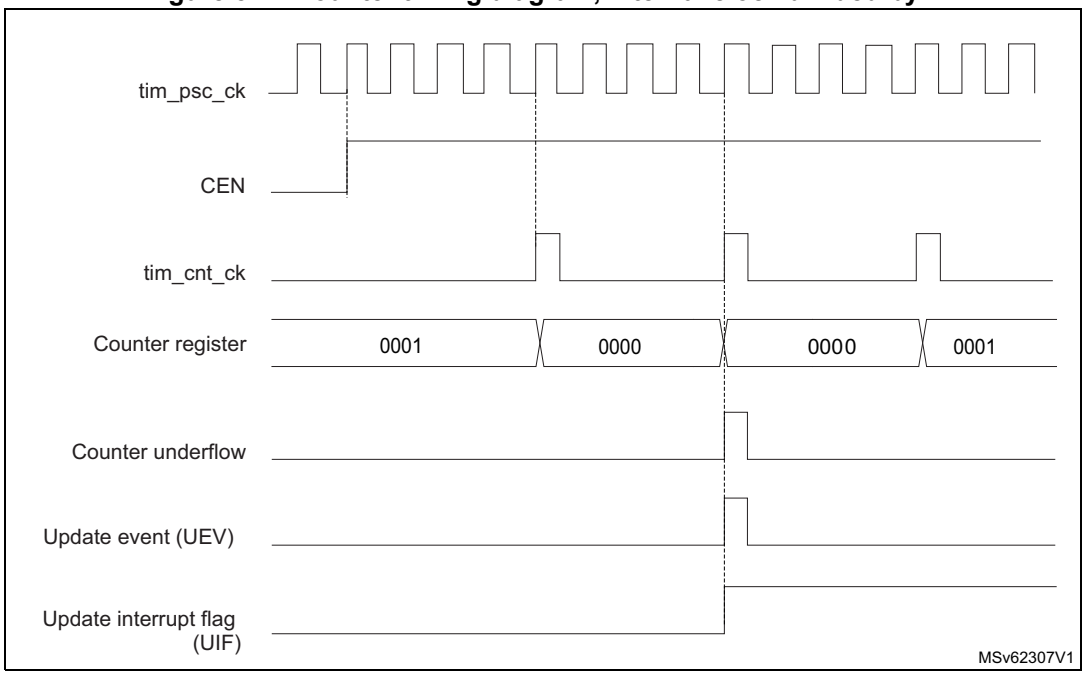


Figure 525. Counter timing diagram, internal clock divided by N

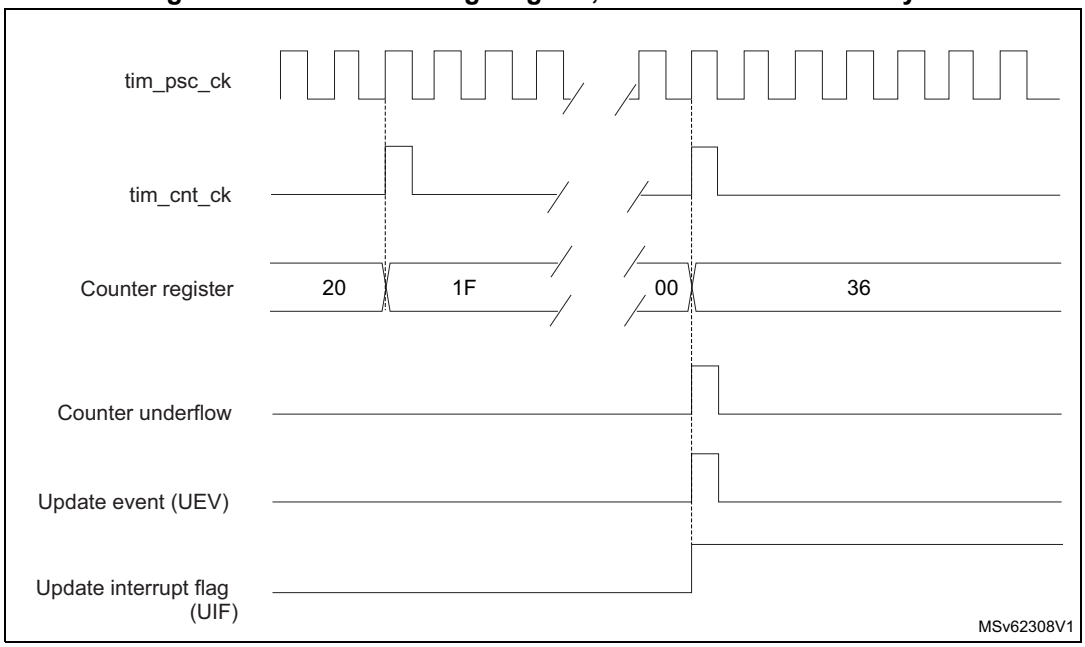
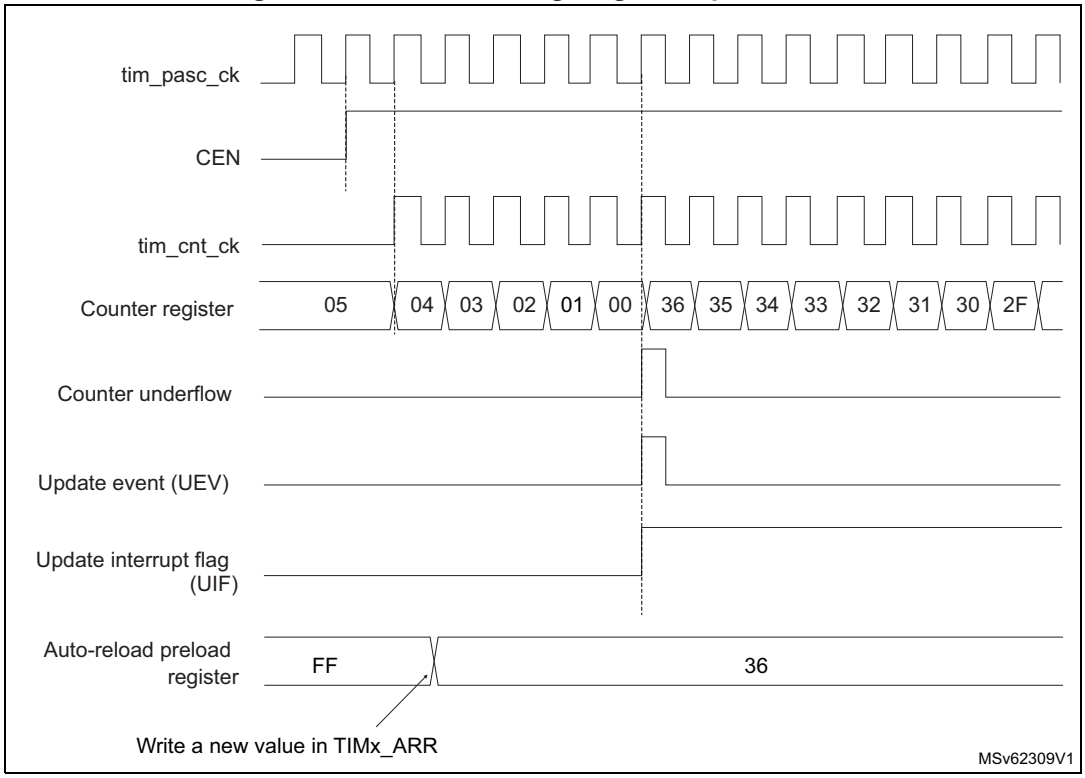


Figure 526. Counter timing diagram, Update event



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-

reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

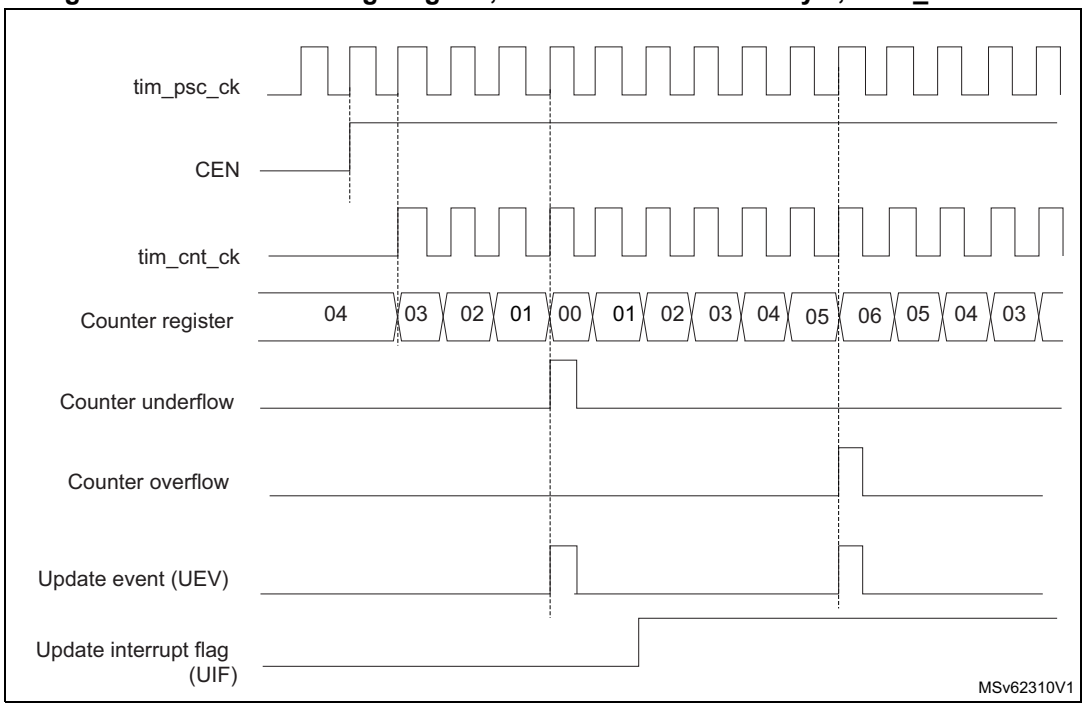
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 527. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6



Note: In the figure above, center-aligned mode 1 is used (for more details refer to Section 36.5.2: TIMx control register 1 (TIMx_CR1)(x = 2 to 5)).

Figure 528. Counter timing diagram, internal clock divided by 2

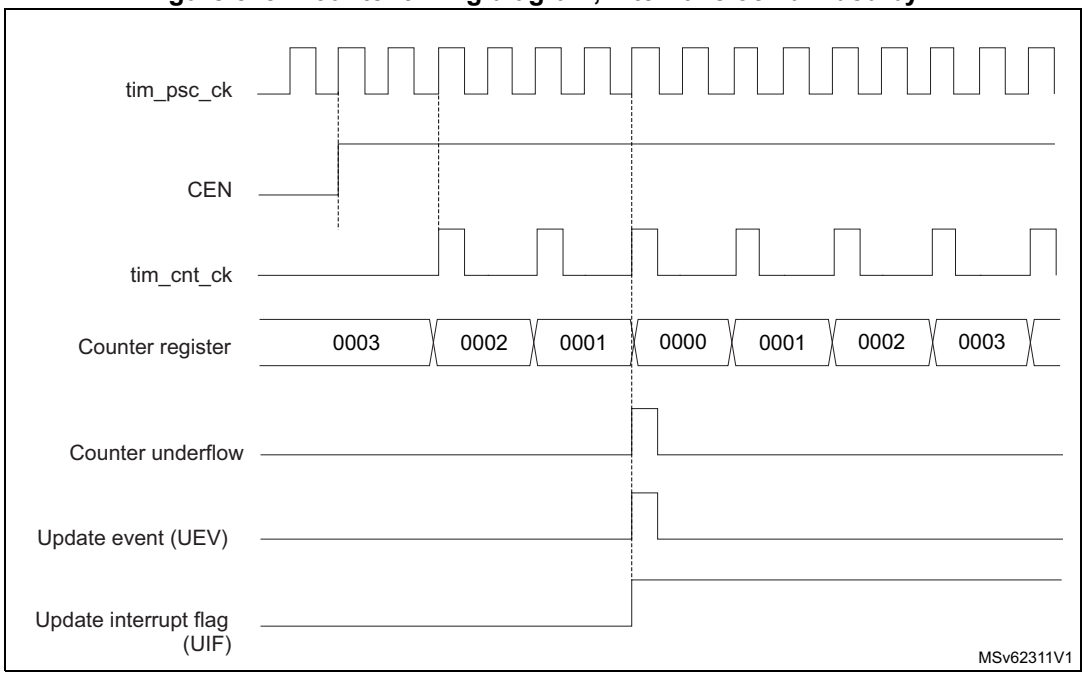
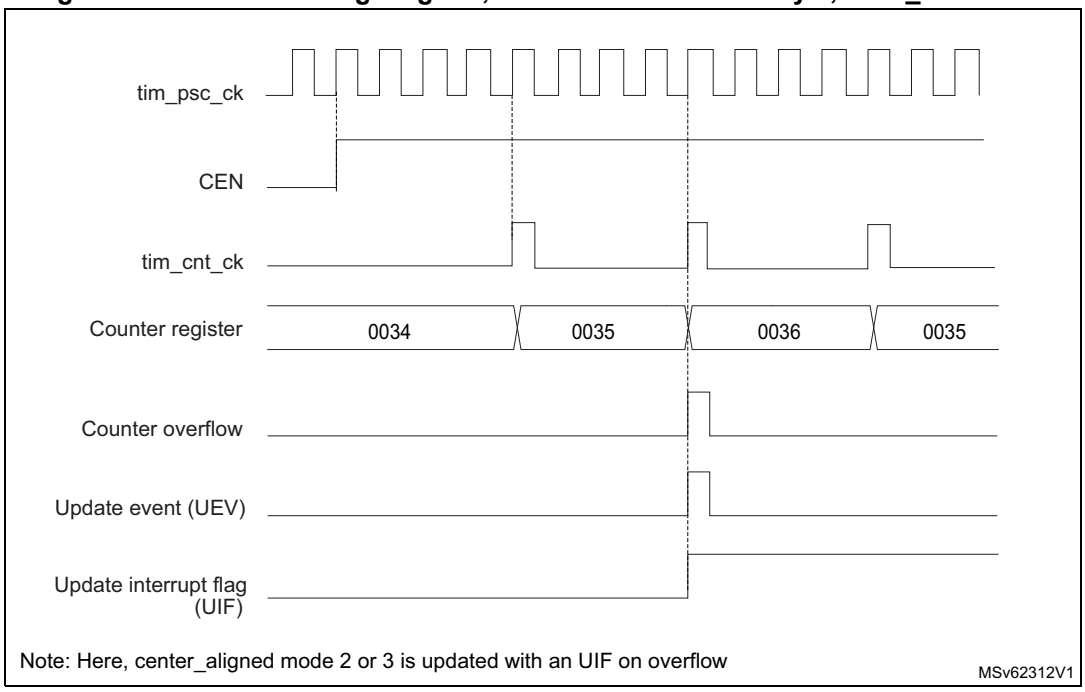


Figure 529. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



Note: In the figure above, center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 530. Counter timing diagram, internal clock divided by N

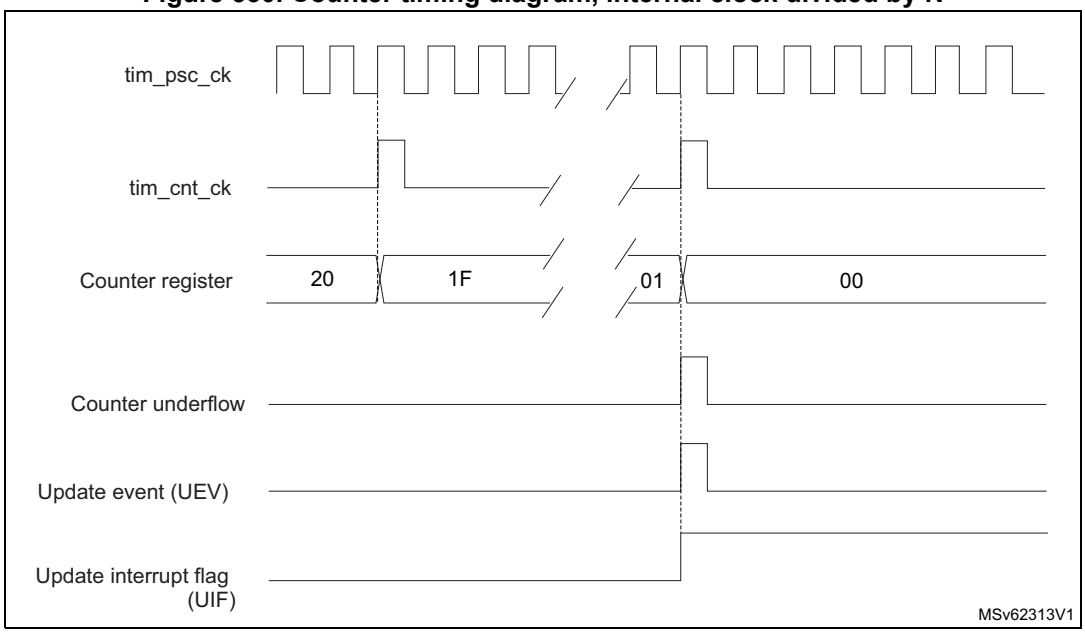


Figure 531. Counter timing diagram, Update event with ARPE=1 (counter underflow)

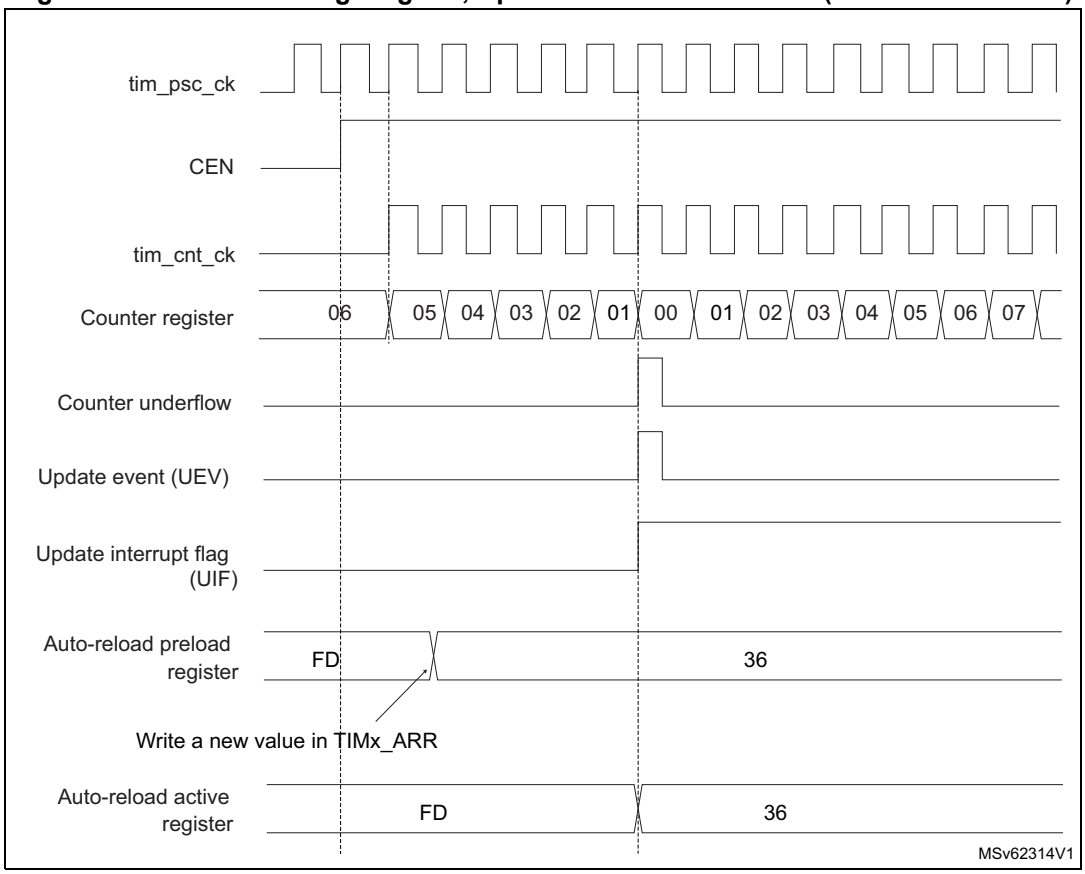
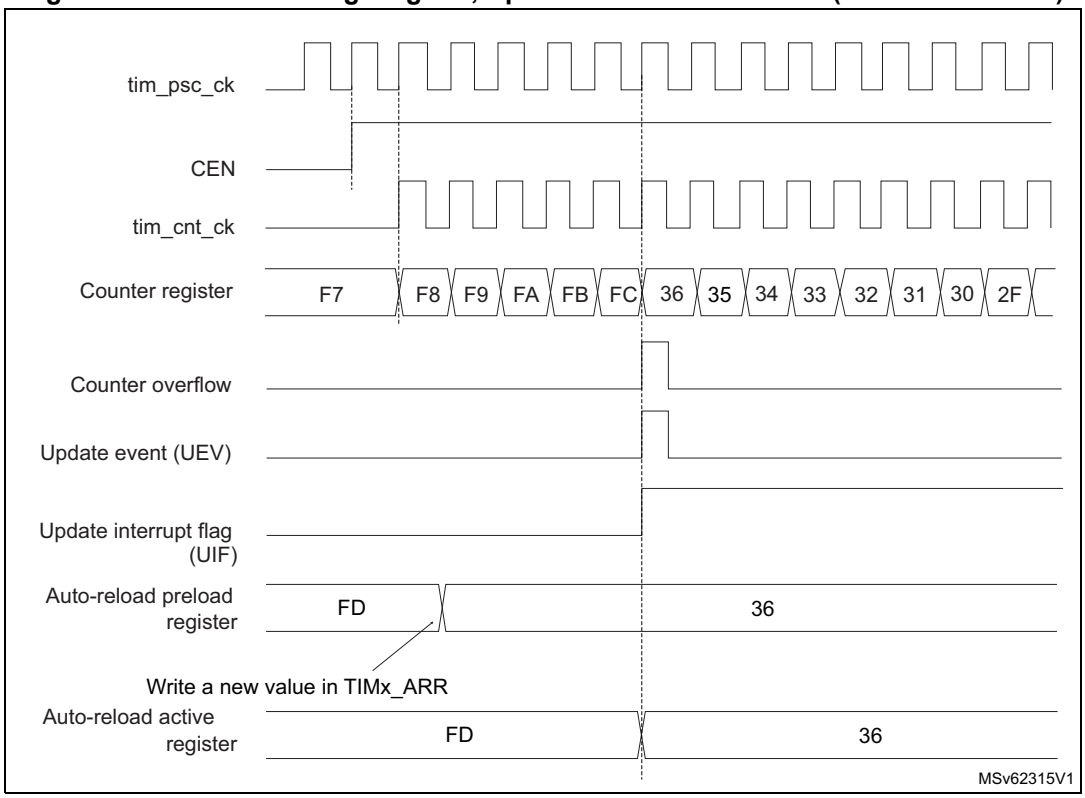


Figure 532. Counter timing diagram, Update event with ARPE=1 (counter overflow)



36.4.5 Clock selection

The counter clock can be provided by the following clock sources:

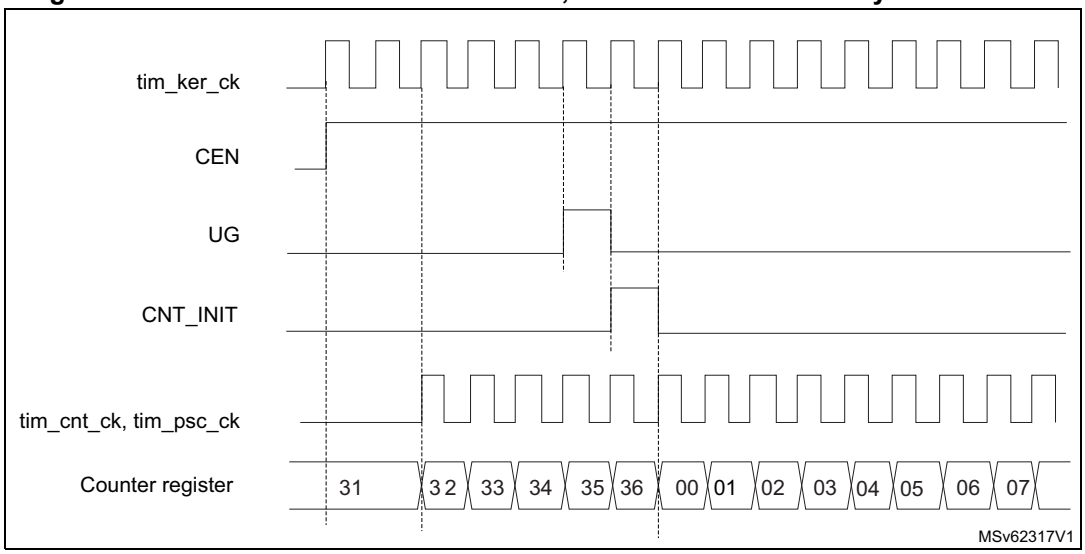
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, Timer 1 can be configured to act as a prescaler for Timer 2. Refer to section *Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 533 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

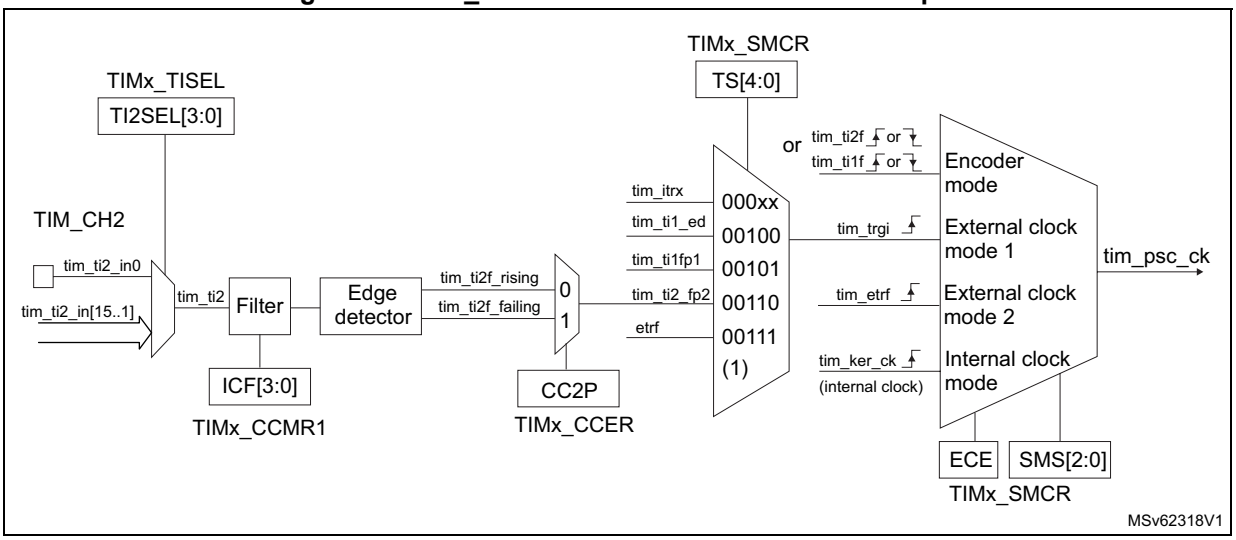
Figure 533. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 534. tim_ti2 external clock connection example



1. In the figure above, codes ranging from 01000 to 11111: tim_itr[15..0].

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= ‘01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

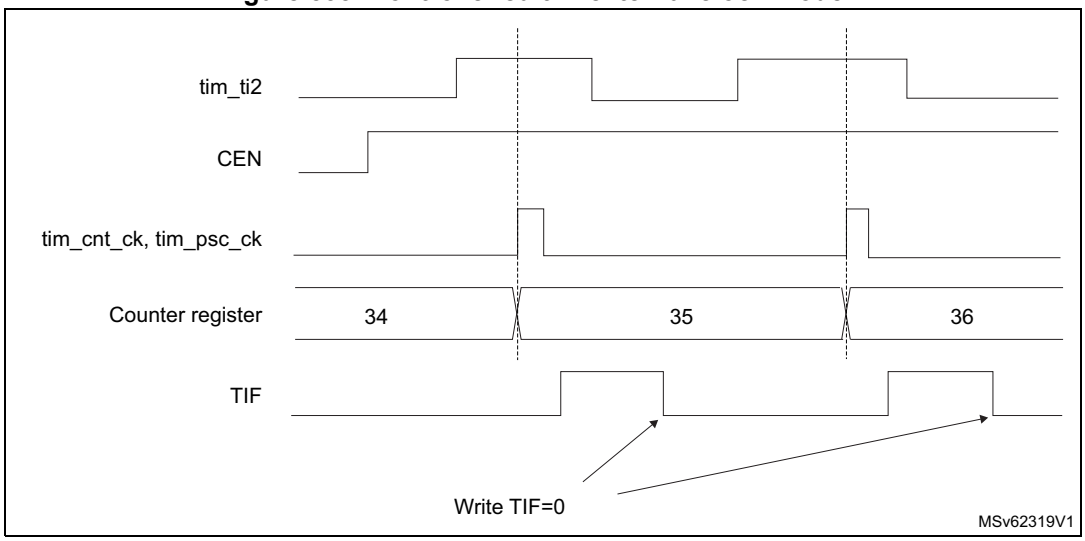
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select tim_ti2 as the input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 535. Control circuit in external clock mode 1



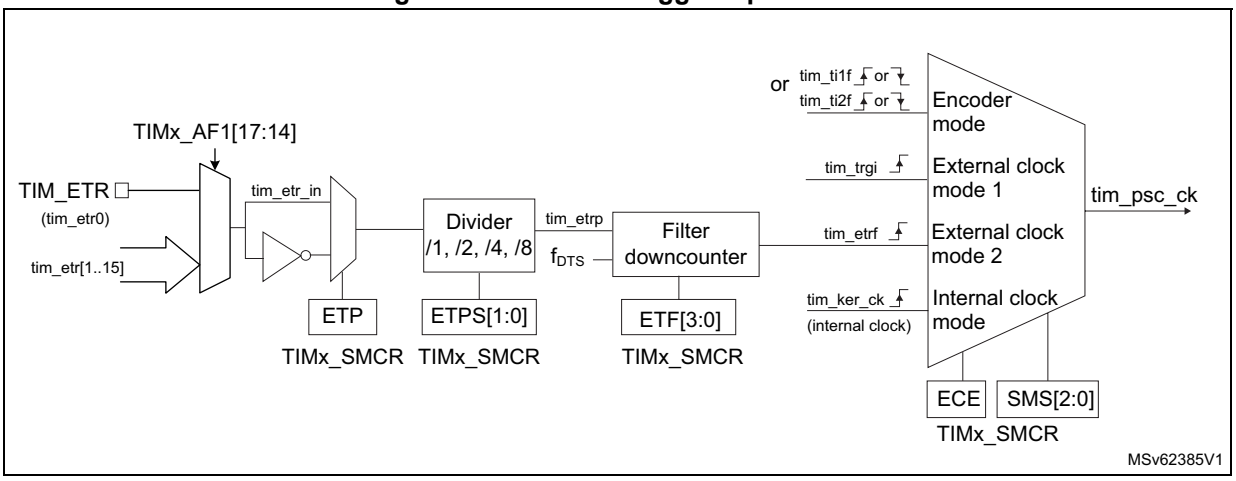
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 536 gives an overview of the external trigger input block.

Figure 536. External trigger input block



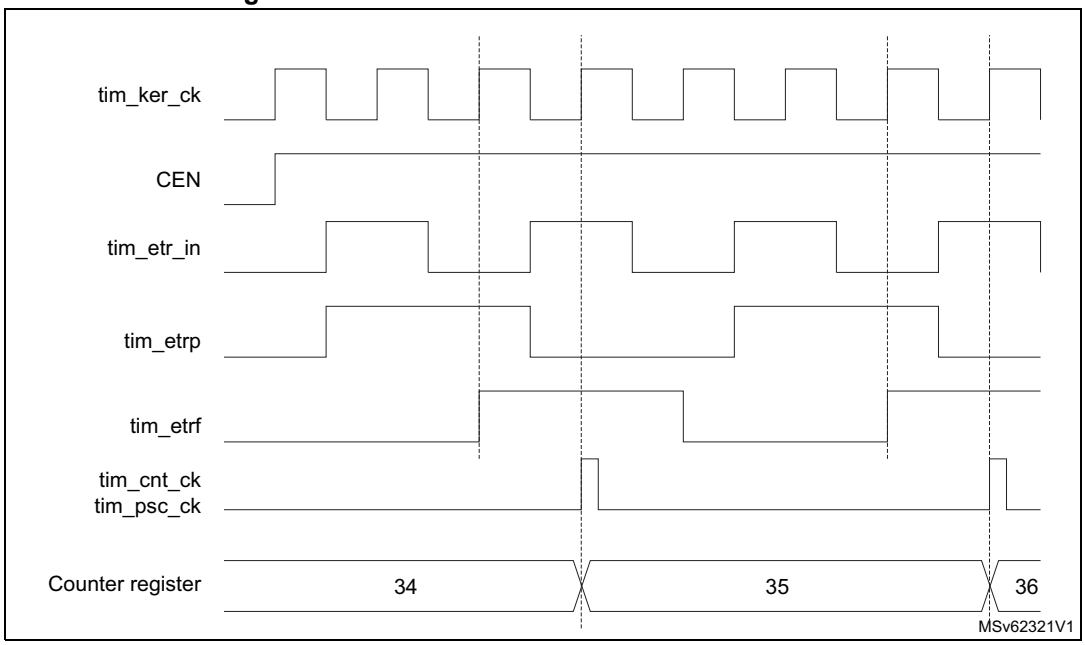
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
4. Select rising edge detection on the tim_etr_in by writing ETP=0 in the TIMx_SMCR register
5. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 537. Control circuit in external clock mode 2



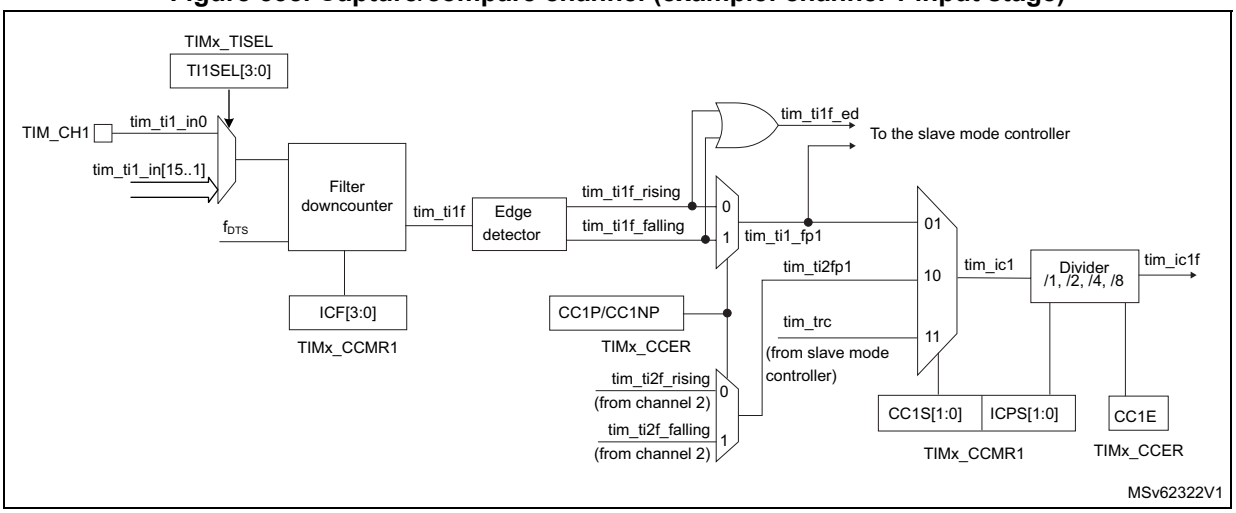
36.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 538. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 539. Capture/compare channel 1 main circuit

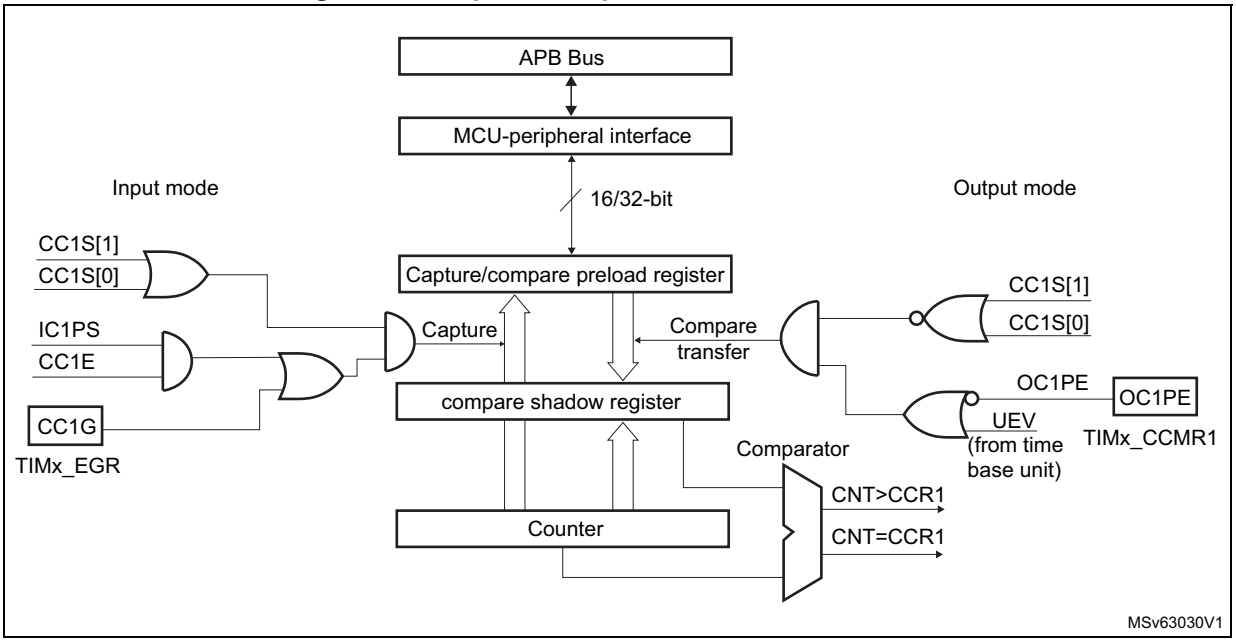
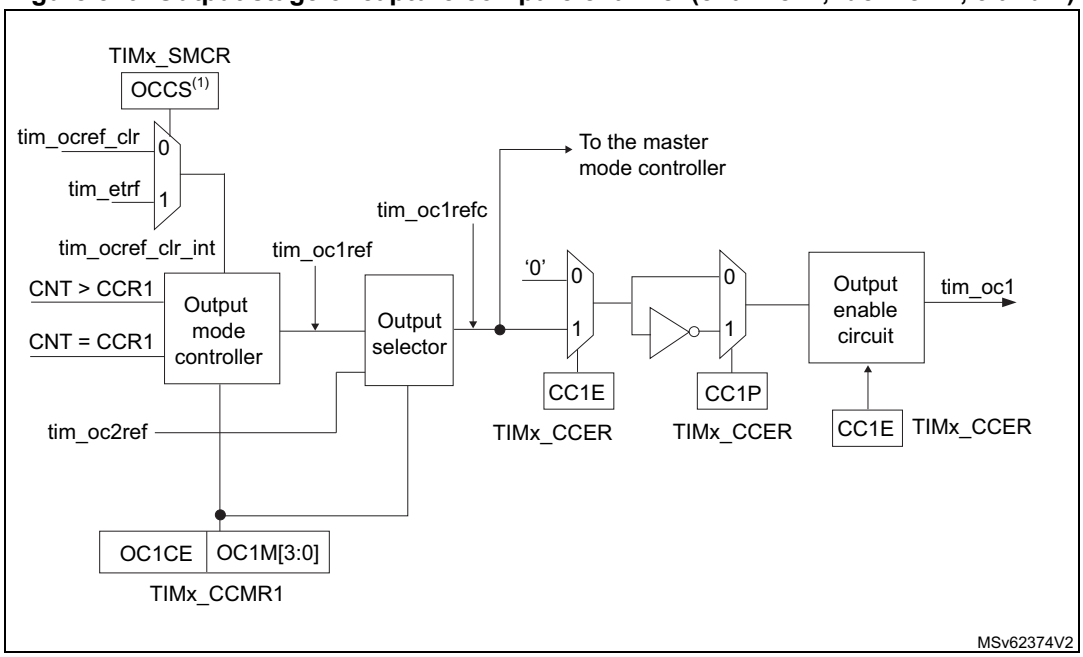


Figure 540. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

36.4.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15..0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_ti1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

36.4.8 PWM input mode

This mode allows to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulse width interval between two consecutive rising and falling edges

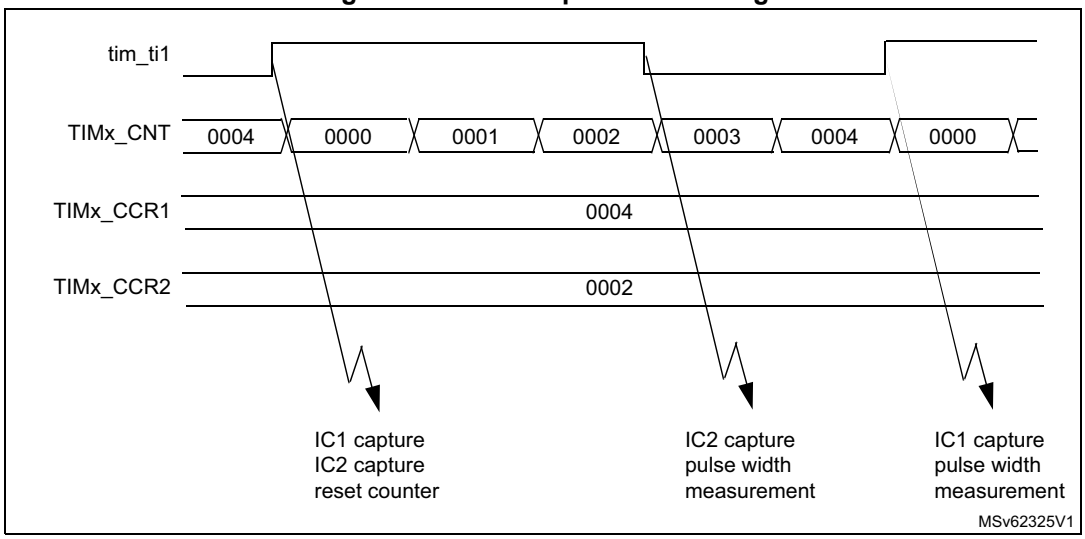
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15..0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ‘0’ and the CC1NP bit to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P bit to ‘1’ and the CC2NP bit to ’0’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1 in the TIMx_CCER register.

Figure 541. PWM input mode timing



Note: The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals.

36.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx gets opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section.

36.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

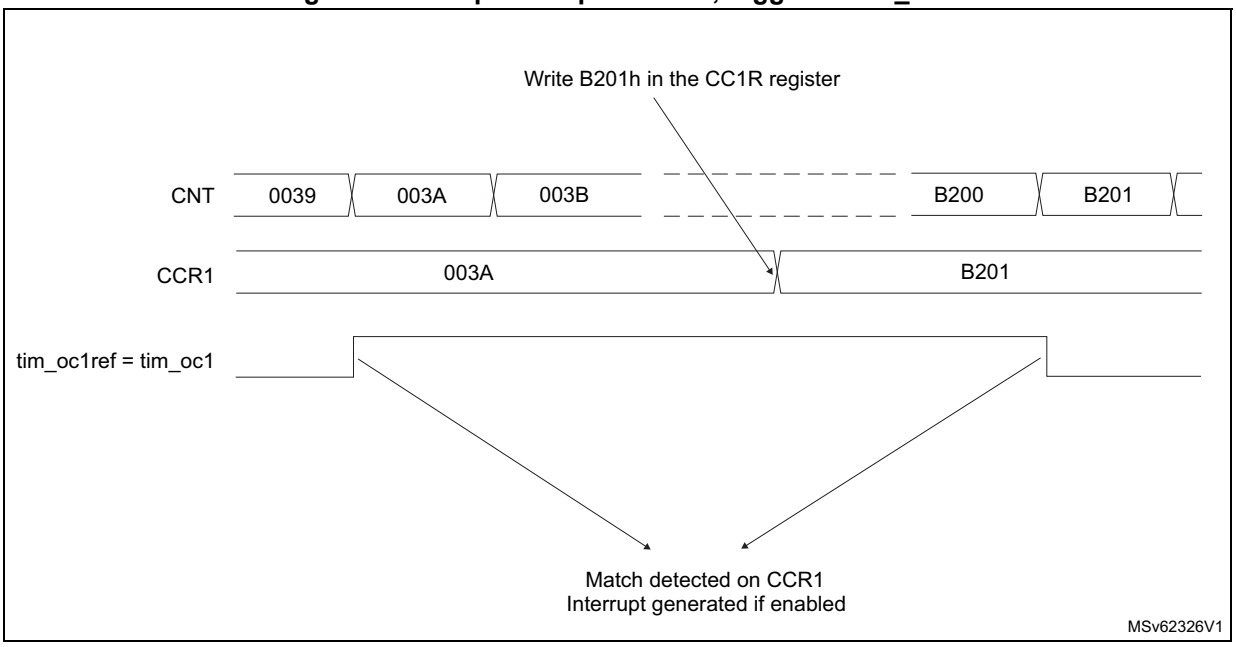
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   b) Write OCxPE = 0 to disable preload register
   c) Write CCxP = 0 to select active high polarity
   d) Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 542*.

Figure 542. Output compare mode, toggle on tim_oc1



36.4.11 PWM mode

Pulse width modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently for each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111). This forces the PWM by software while the timer is running.

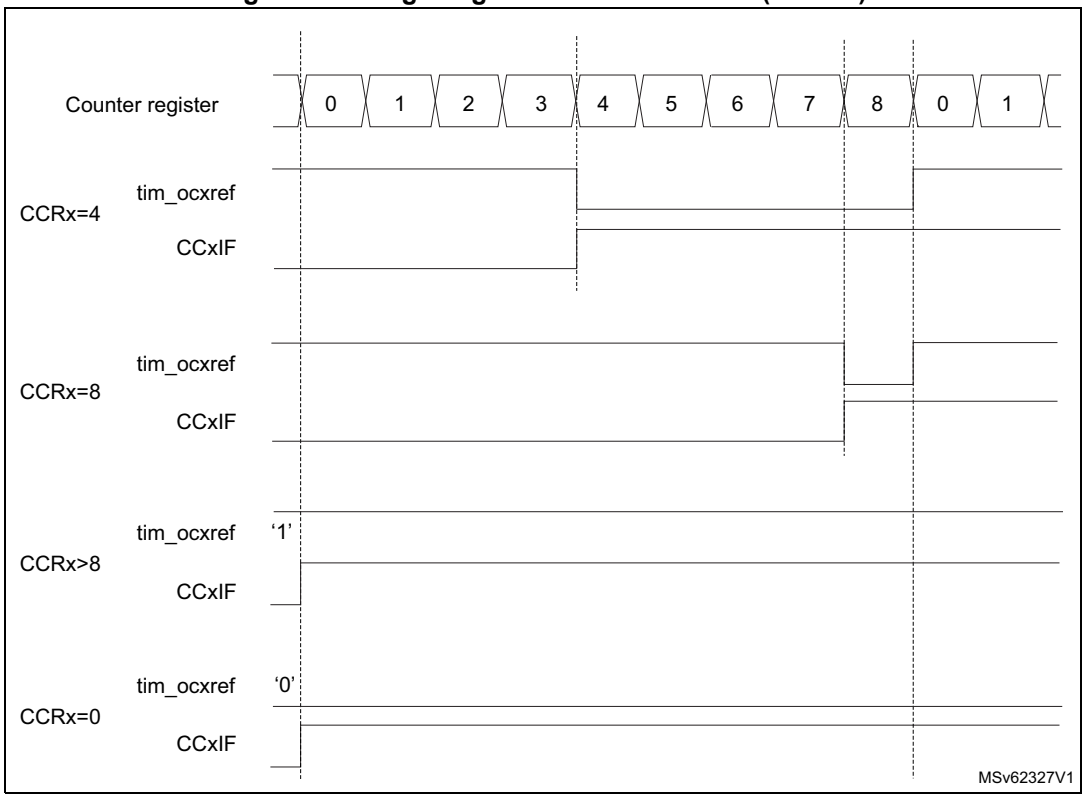
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

Upcounting configuration is active when the DIR bit in the TIMx_CR1 register is low. Refer to section *Upcounting mode*.

In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then tim_ocxref is held at ‘1. If the compare value is 0 then tim_ocxref is held at ‘0. *Figure 543* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 543. Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration is active when DIR bit in TIMx_CR1 register is high. Refer to section *Downcounting mode*.

In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

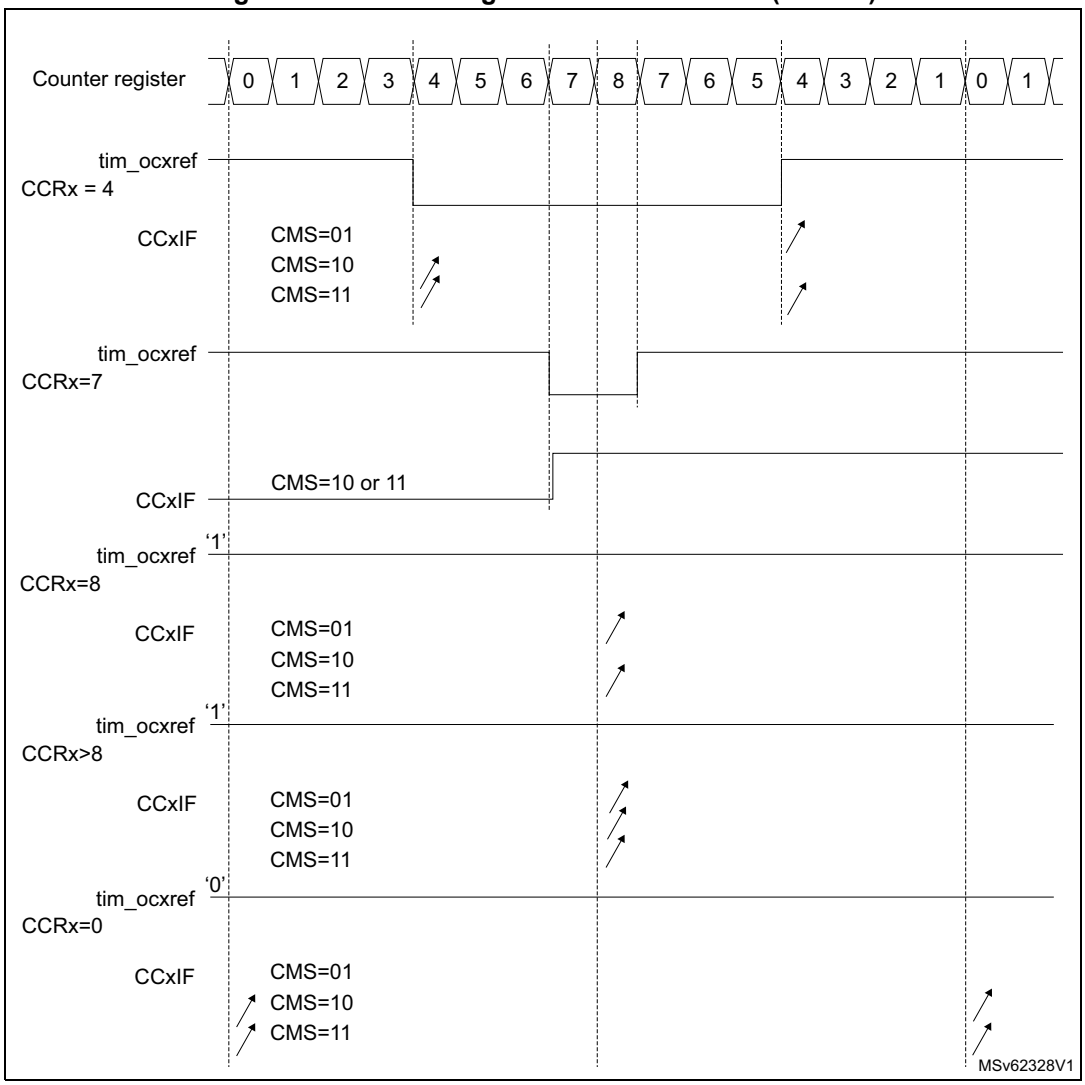
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to section *Center-aligned mode (up/down counting)*.

Figure 544 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 544. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

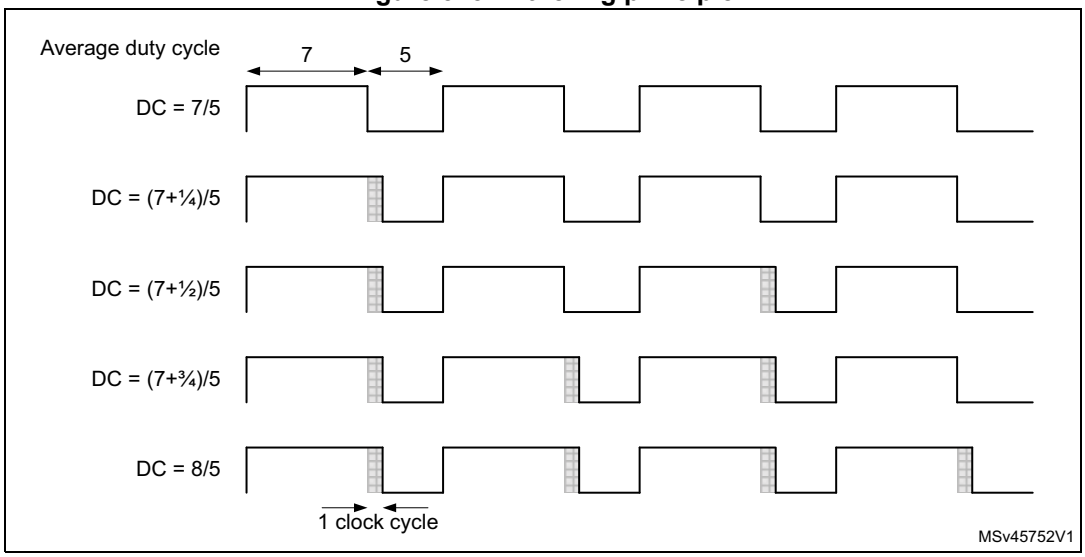
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 545* presents the dithering principle applied to 4 consecutive PWM cycles.

Figure 545. Dithering principle

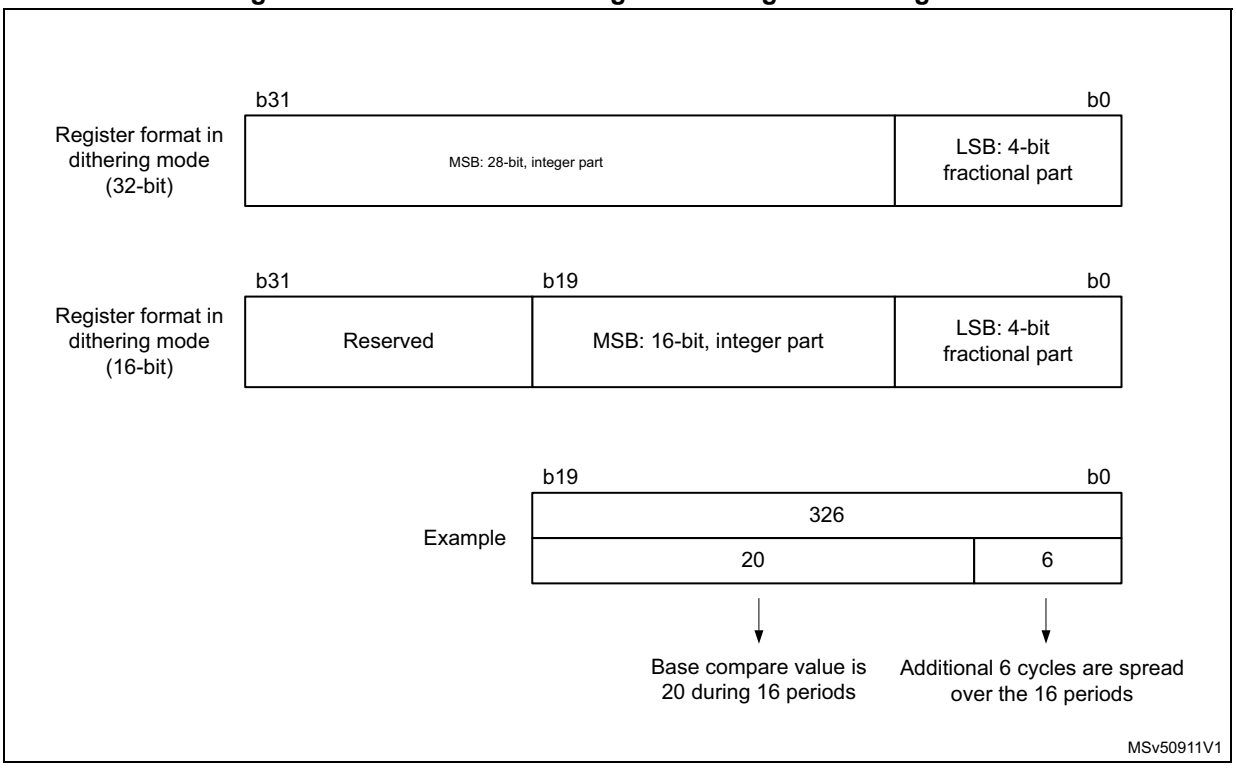


When the dithering mode is enabled, the register coding is changed as follows (refer to *Figure 546* for example):

- The 4 LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by 4 places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The ARR and CCR values are updated automatically if the DITHEN bit is set / reset (for instance, if ARR= 0x05 with DITHEN=0, it is updated to ARR = 0x50 with DITHEN=1). The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The DITHEN bit must be reset
4. The CCIF flags must be cleared
5. The CEN bit can be set (eventually with ARPE = 1)

Figure 546. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown in the *Figure 547* and *Figure 548*, the dithering mode allows to increase the PWM resolution.

Figure 547. PWM resolution vs frequency (16-bit mode)

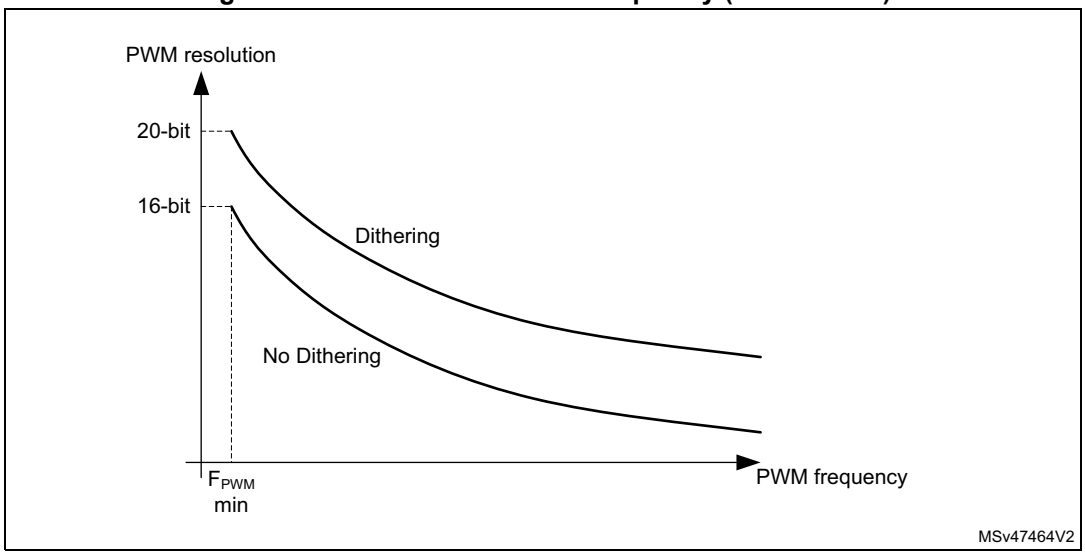
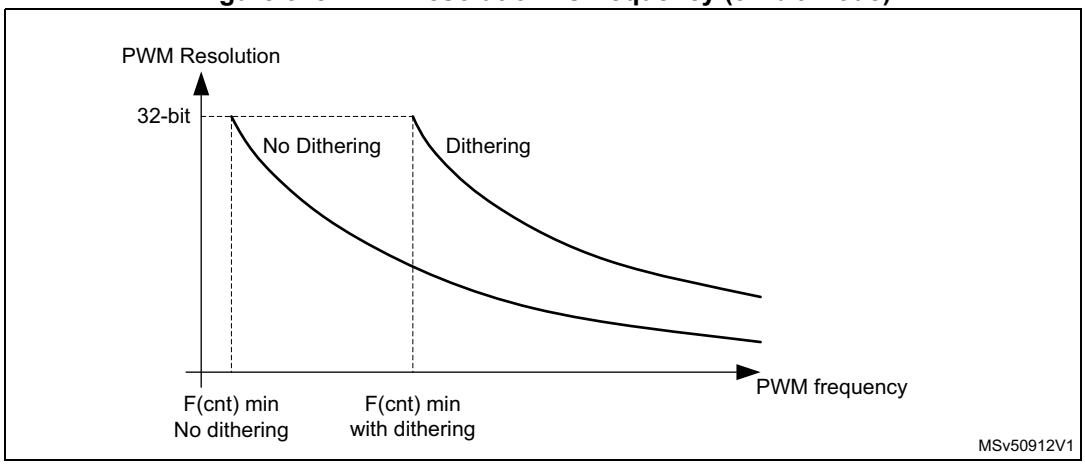
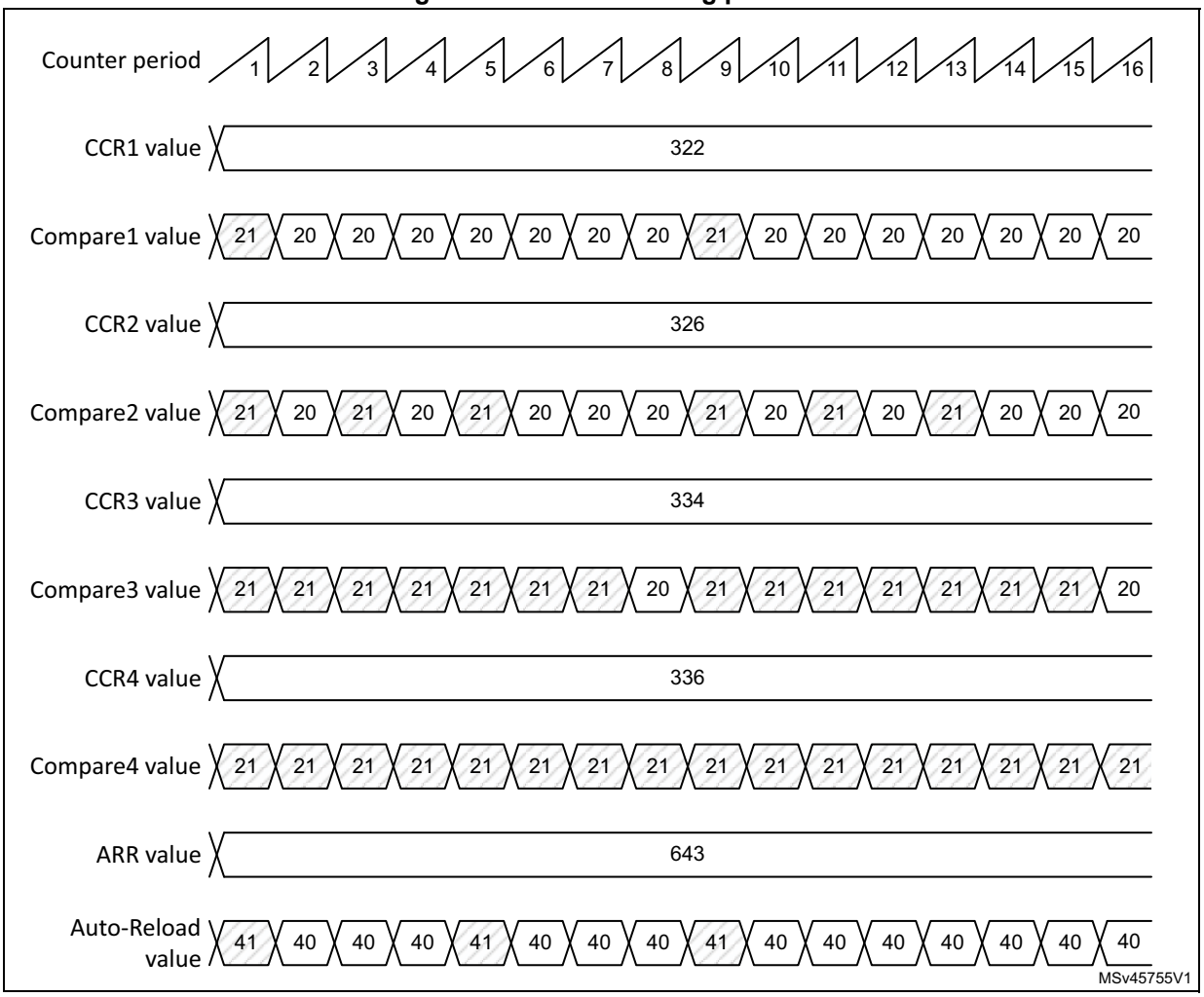


Figure 548. PWM resolution vs frequency (32-bit mode)



The duty cycle and / or period changes are spread over 16 consecutive periods, as described in the *Figure 549*.

Figure 549. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 427*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 427. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to '00'). In this case, the dithering pattern is applied over 8 consecutive PWM periods, considering the up and down counting phases as shown in the *Figure 550*.

Figure 550. Dithering effect on duty cycle in center-aligned PWM mode

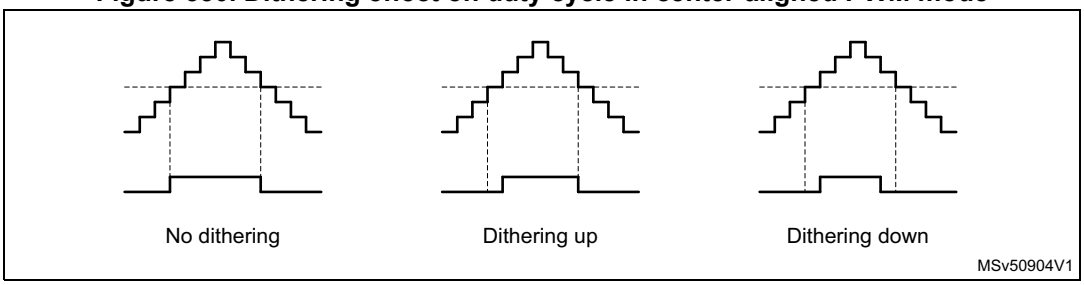


Table 428 shows how the dithering pattern is added in center-aligned PWM mode.

Table 428. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

36.4.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2.
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4.

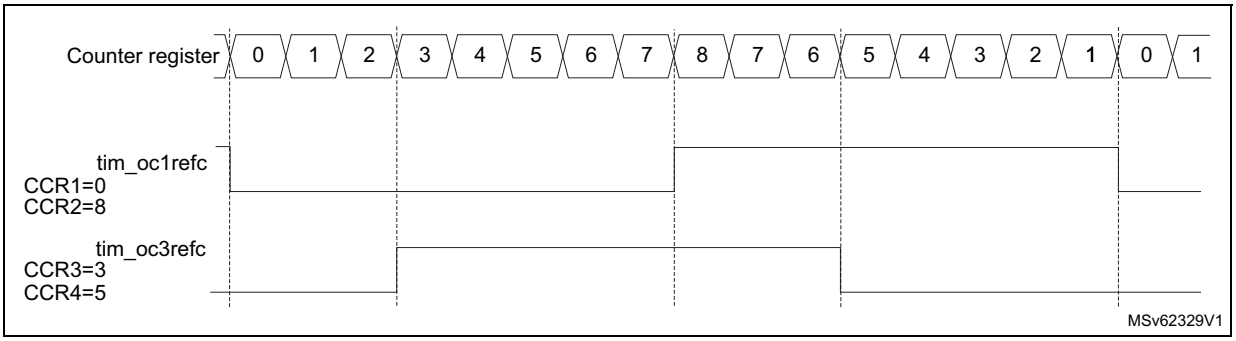
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if a tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or a tim_oc2refc signal resulting from asymmetric PWM mode 2.

Figure 551 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 2).

Figure 551. Generation of 2 phase-shifted PWM signals with 50% duty cycle



36.4.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

– tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2.

– tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4.

Combined PWM mode can be selected independently for two channels (one tim_ocx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

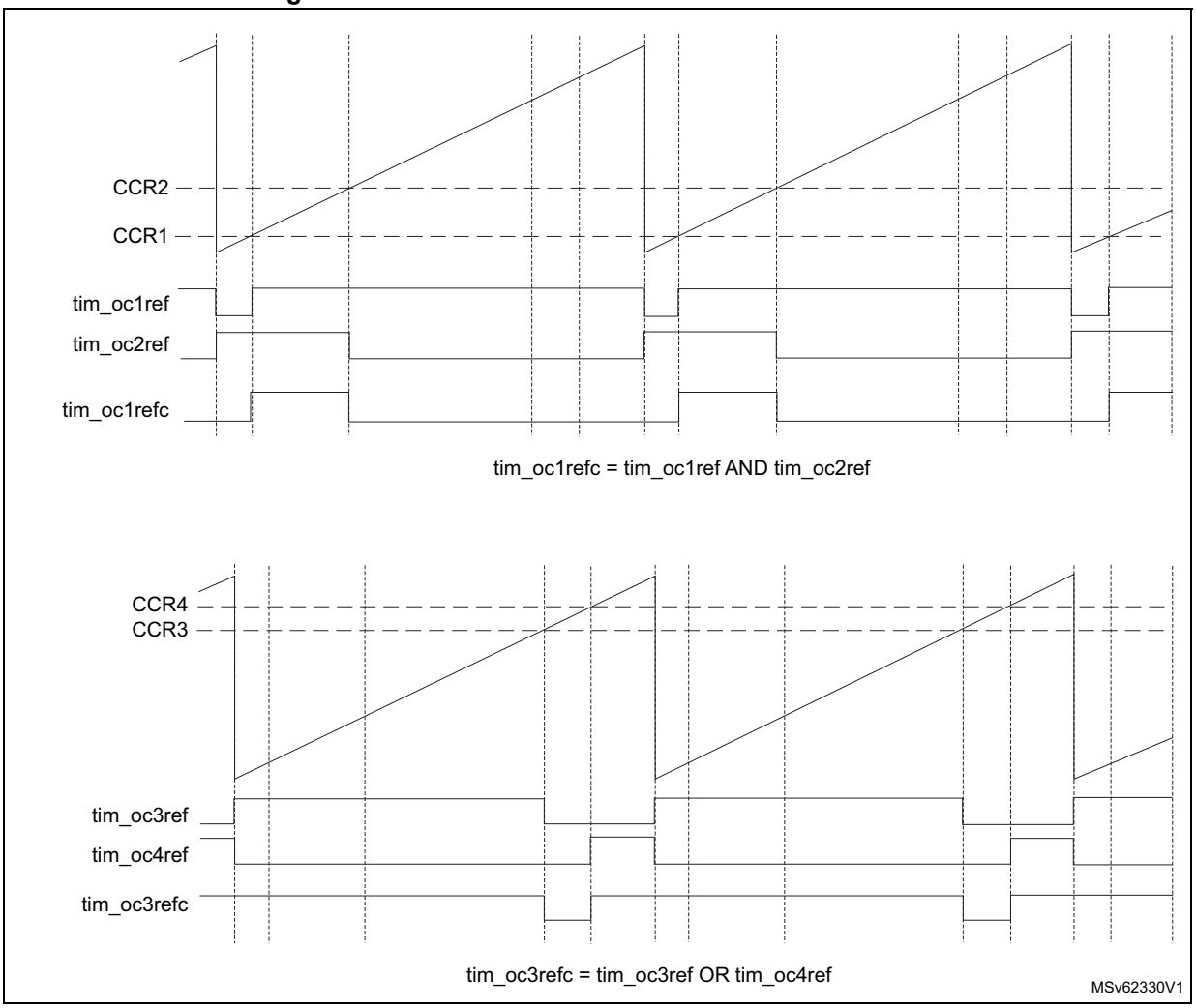
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 552 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2
- Channel 2 is configured in PWM mode 1
- Channel 3 is configured in Combined PWM mode 1
- Channel 4 is configured in PWM mode 2

Figure 552. Combined PWM mode on channels 1 and 3



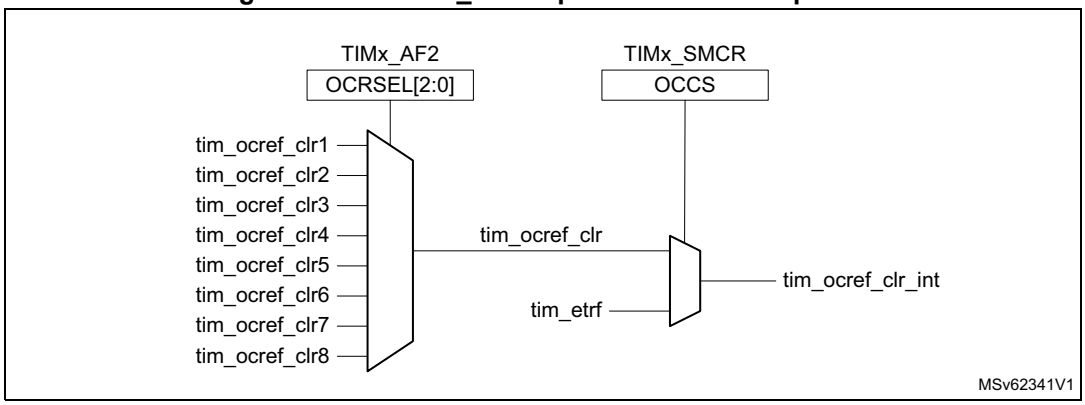
36.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 36.3: TIM2/TIM3/TIM4/TIM5 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7..0] inputs, using the OCRSEL[2:0] bit field in the TIMx_AF2 register, as shown in *Figure 553*.

Figure 553. OCREF_CLR input selection multiplexer



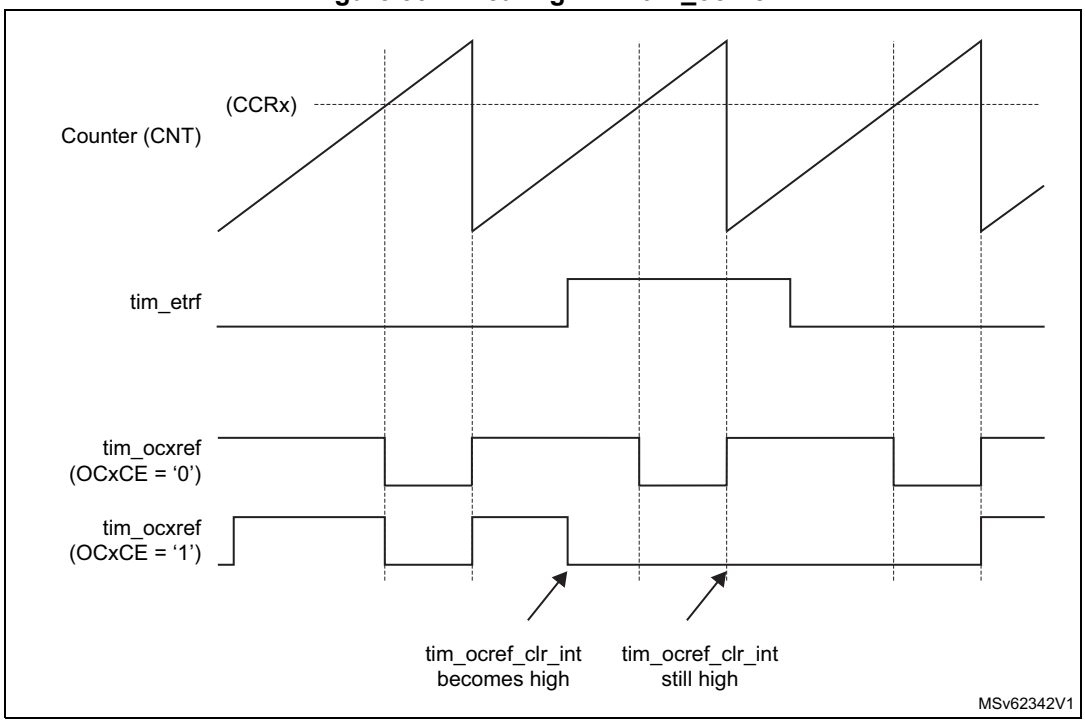
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 554 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 554. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

36.4.15 One-pulse mode

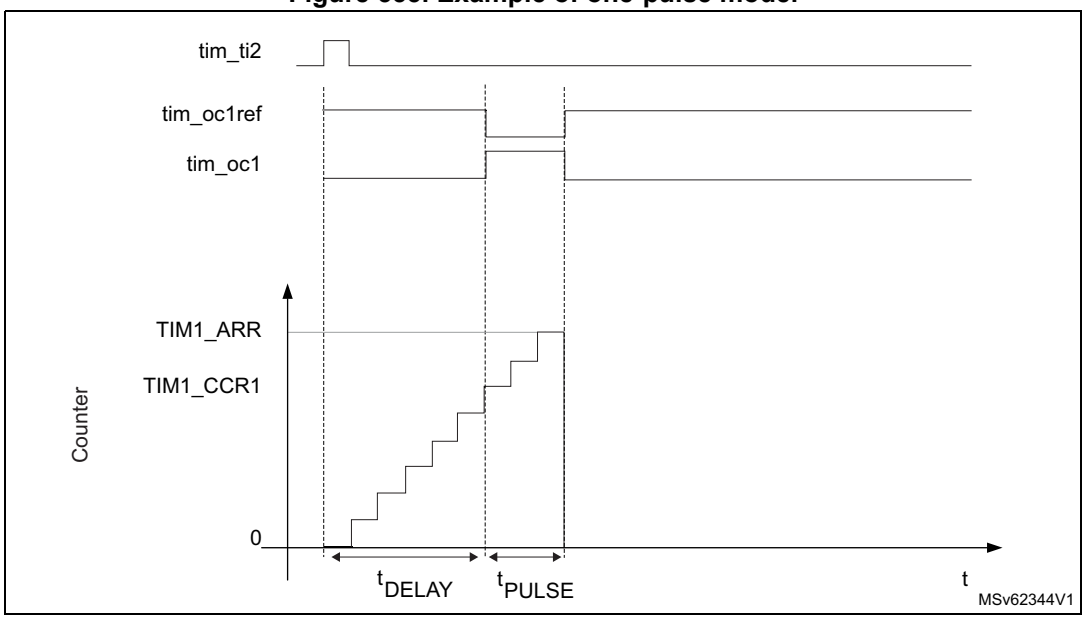
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT<CCRx ≤ ARR (in particular, 0<CCRx),

Figure 555. Example of one-pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15..0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS=00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one wants to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

36.4.16 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 36.4.15: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

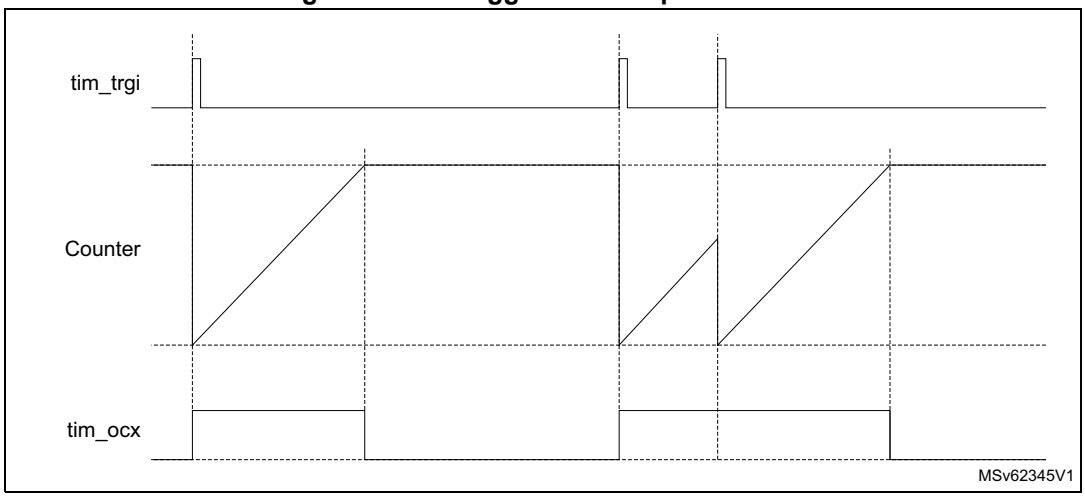
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

Note: In retriggerable one pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 556. Retriggerable one pulse mode



36.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown in the *Figure 557*.

Figure 557. Pulse generator circuitry

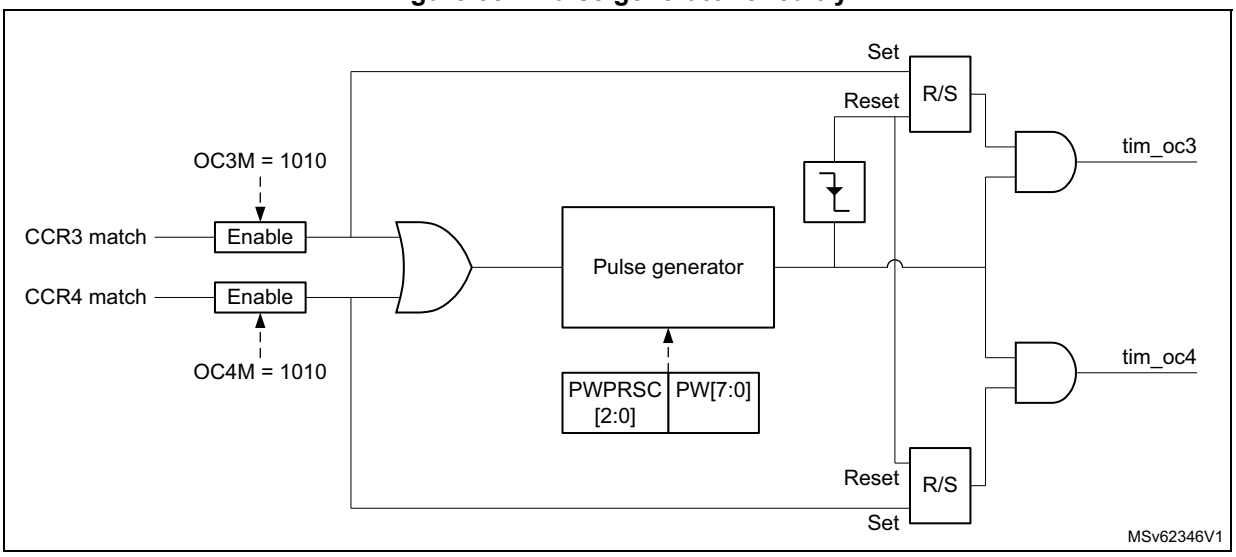
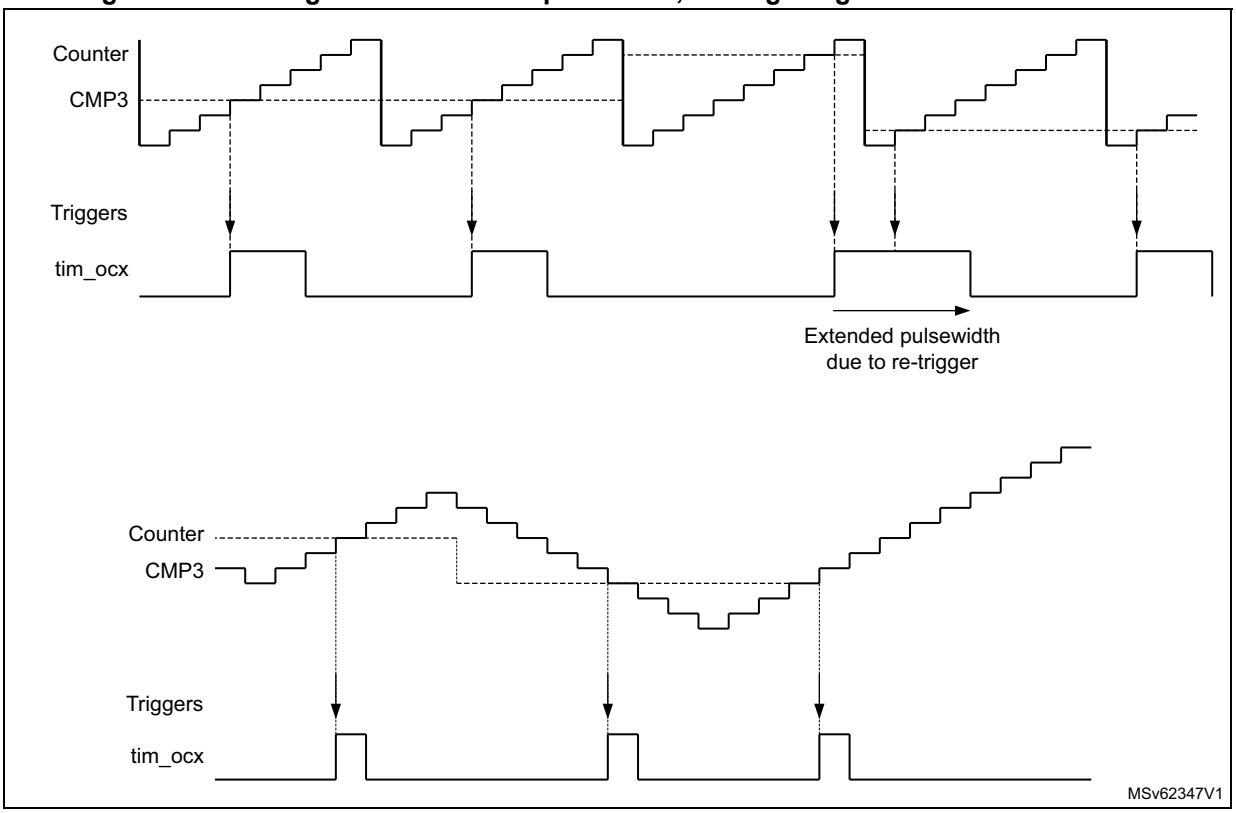


Figure 558 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 558. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bit fields in TIMx_CCMR2 register.

The pulse width is programmed using the PW[7:0] bit field in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

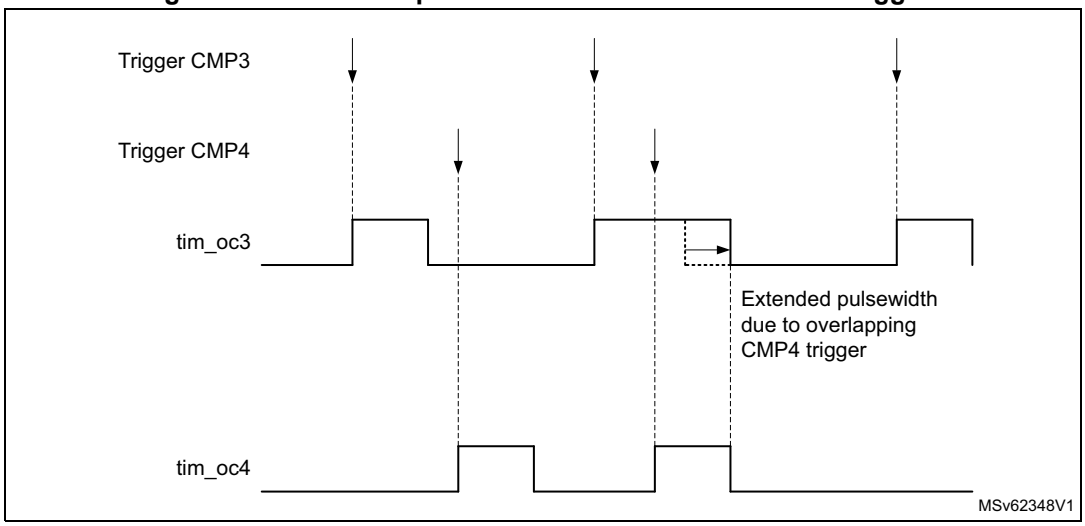
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is re-triggerable: a new trigger while the pulse is on-going causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the 1st arriving trigger is extended (because of the re-trigger), while the pulse width of the last arriving trigger is correct (as shown in the Figure 559).

Figure 559. Extended pulse width in case of concurrent triggers



36.4.18 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS='0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS=0010 if it is counting on tim_ti2 edges only and SMS=0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 429*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1=tim_ti1 if not filtered and not inverted, tim_ti2fp2=tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 429. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, may be connected to the external trigger input and triggers a counter reset.

Figure 560 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1)
- CC2S= 01 (TIMx_CCMR2 register, tim_ti2fp2 mapped on tim_ti2)
- CC1P and CC1NP = '0' (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1=tim_ti1)
- CC2P and CC2NP = '0' (TIMx_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2=tim_ti2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

Figure 560. Example of counter operation in encoder interface mode

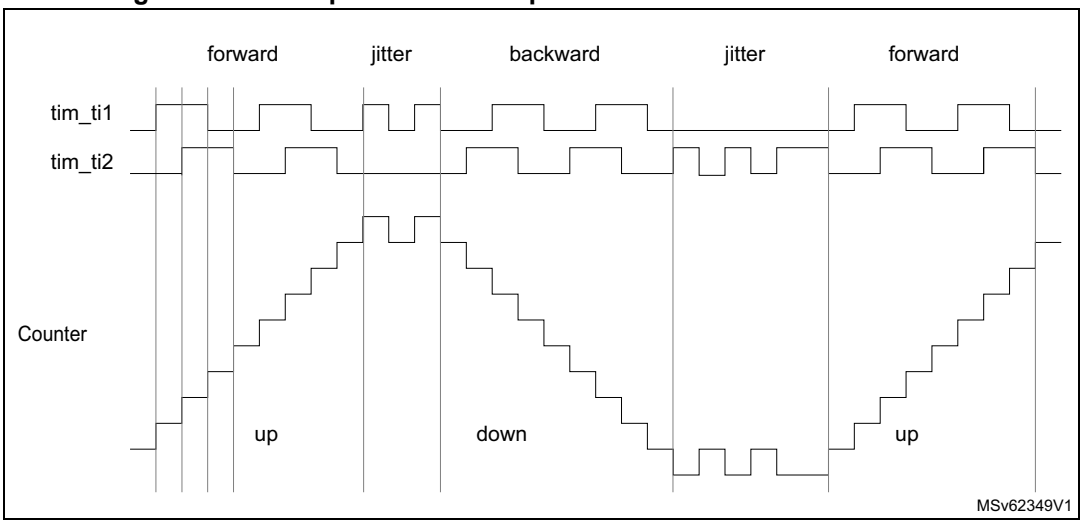


Figure 561 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 561. Example of encoder interface mode with tim_ti1fp1 polarity inverted

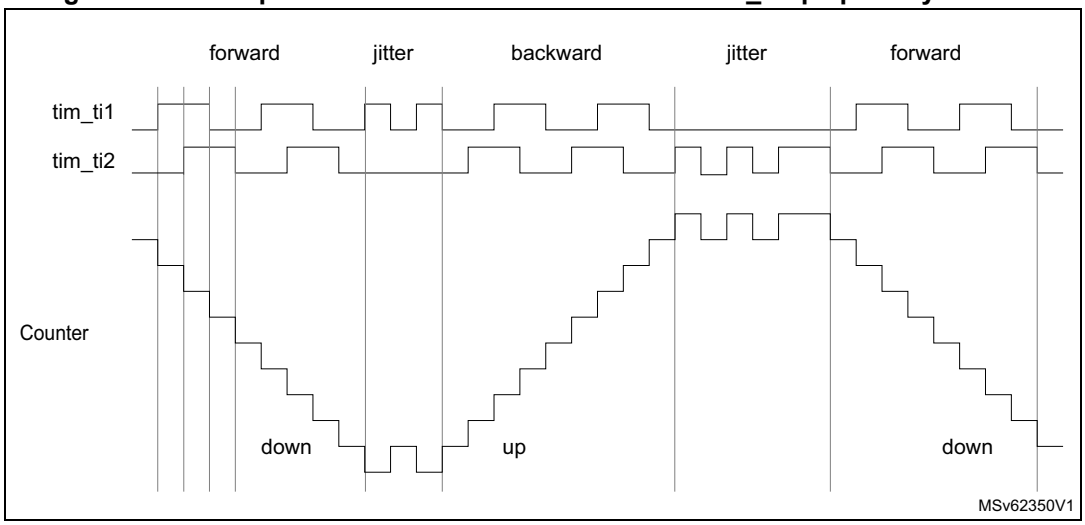
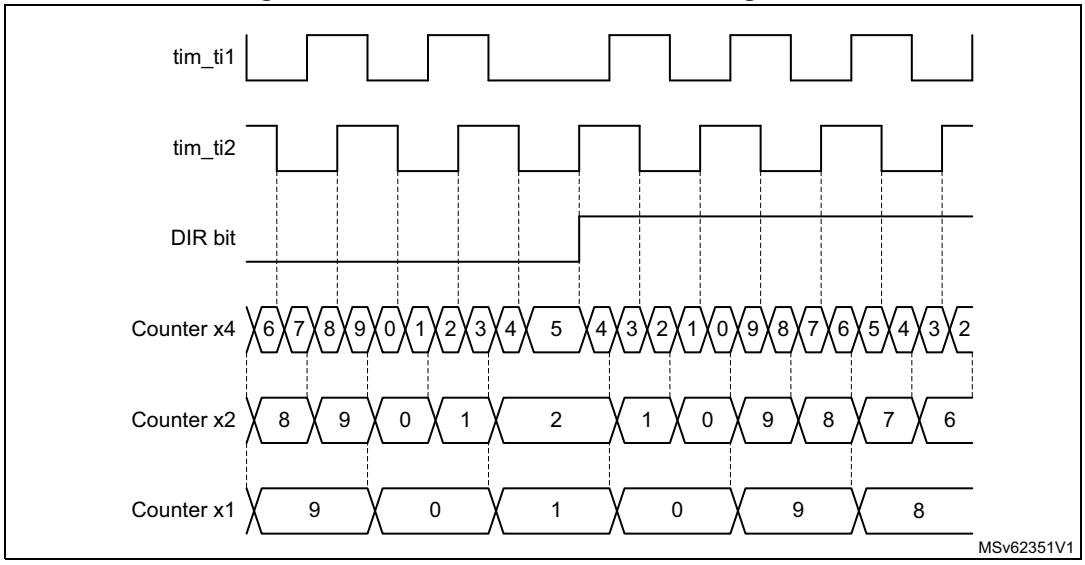


Figure 562 shows the timer counter value during a speed reversal, for various counting modes.

Figure 562. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support to other types of encoders.

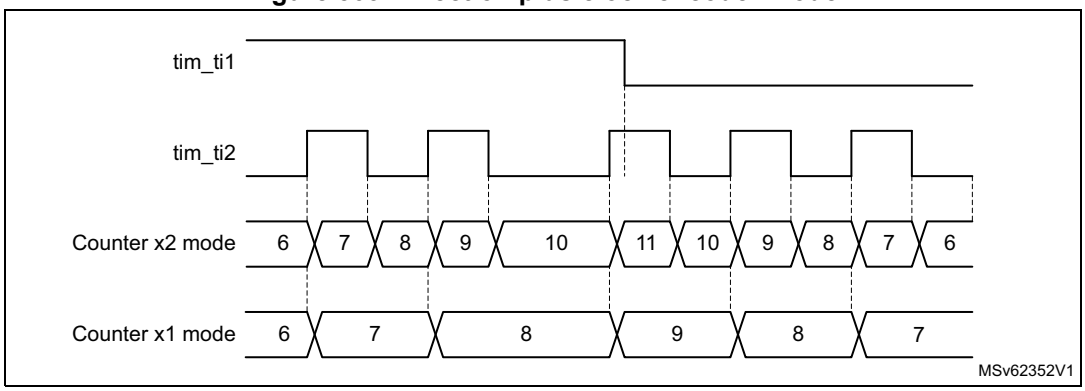
In the "clock plus direction" mode shown in *Figure 563*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bit field in the TIMx_SMCR register, as follows:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock.
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 563. Direction plus clock encoder mode



Directional Clock encoder mode

In the “directional clock” mode in *Figure 564*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bit field in the TIMx_SMCR register, as follows:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 564*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 565*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 564*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 565*).

Figure 564. Directional clock encoder mode (CC1P = CC2P = 0)

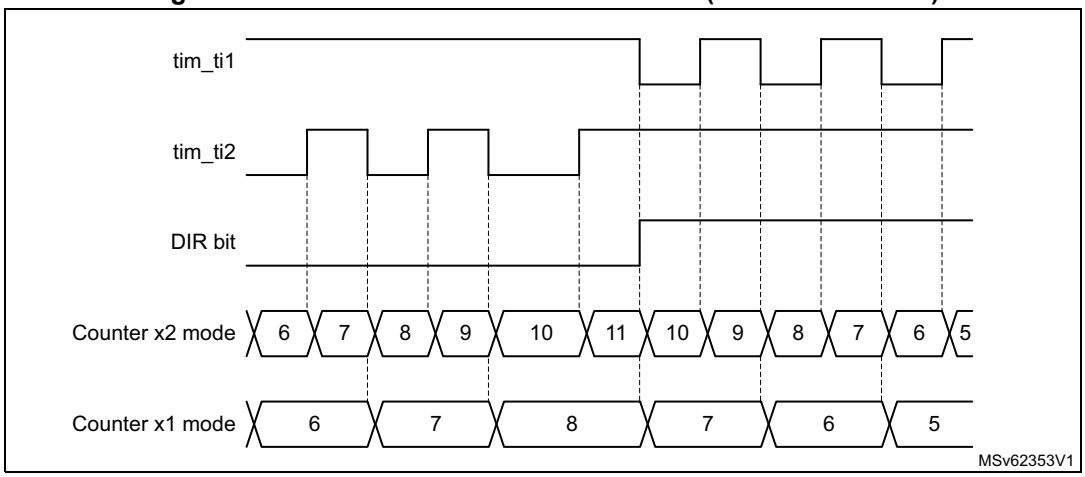


Figure 565. Directional clock encoder mode (CC1P = CC2P = 1)

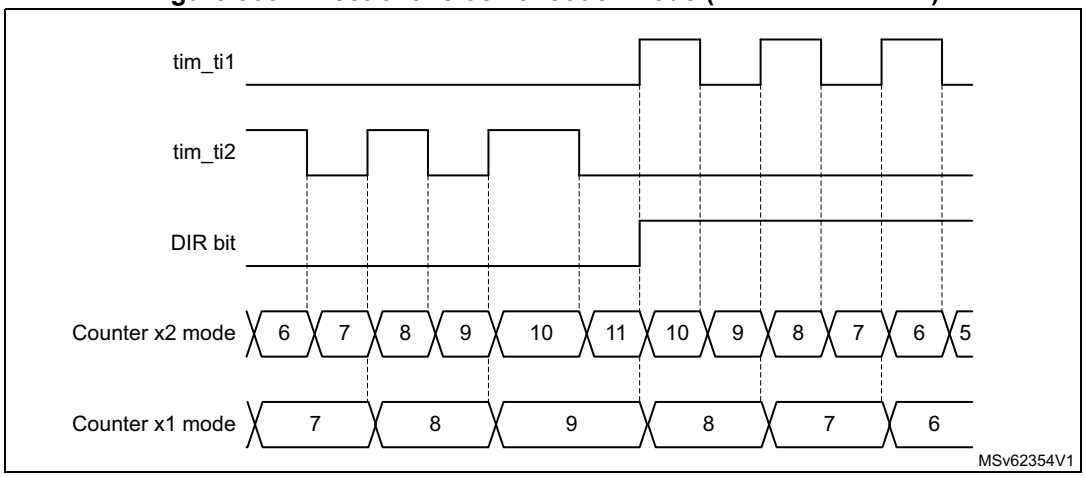


Table 430 details how the directional clock mode operates, for any input transition.

Table 430. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP=0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP=1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP=0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP=1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index Input

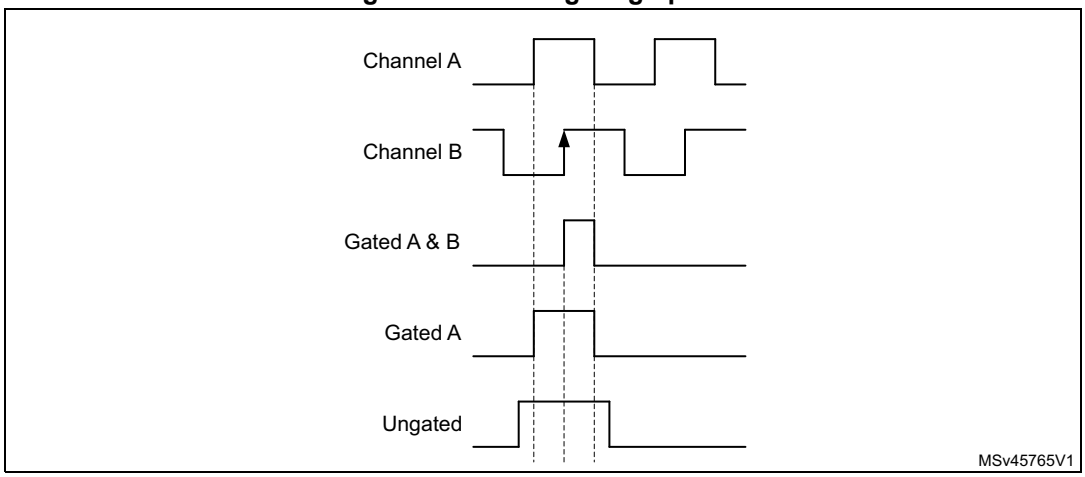
The counter can be reset by an Index signal coming from the encoder, indicating an absolute reference position. The Index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bit field has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Commercially available encoders are proposed with several options for index pulse conditioning, as per the *Figure 566*:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.

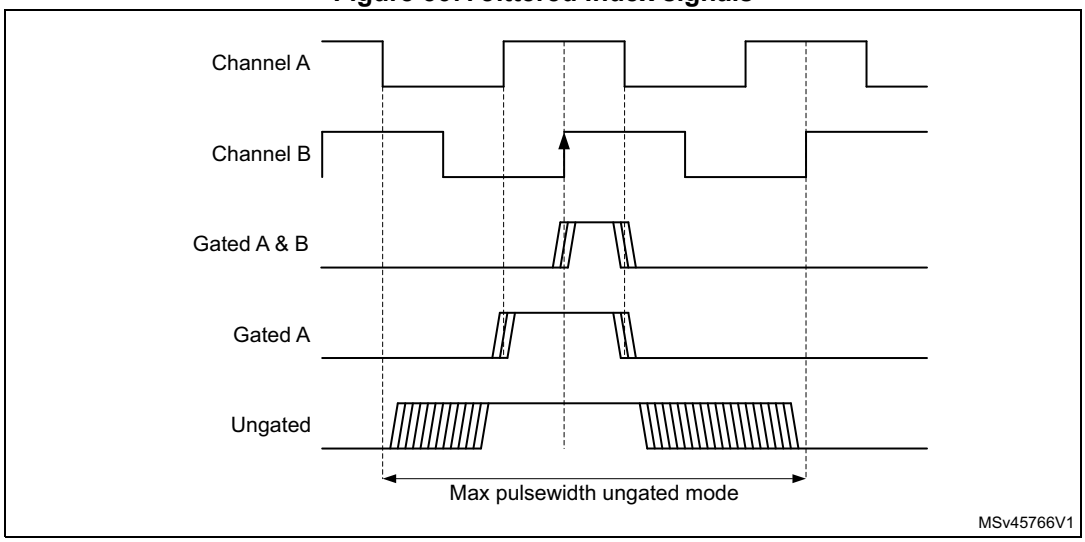
Figure 566. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown in *Figure 567*.

In ungated mode, the signal must be strictly 2 encoder periods. If the pulse width is greater than or equal to 2 encoder period, the counter is reset multiple times.

Figure 567. Jittered Index signals



The timer supports the 3 gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (that are channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bit field in the TIMx_ECR register.

The Index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 568* shows at which position is the index generated, for a simplistic example (an encoder providing 4 edges par mechanical rotation).

Figure 568. Index generation for IPOS[1:0] = 11

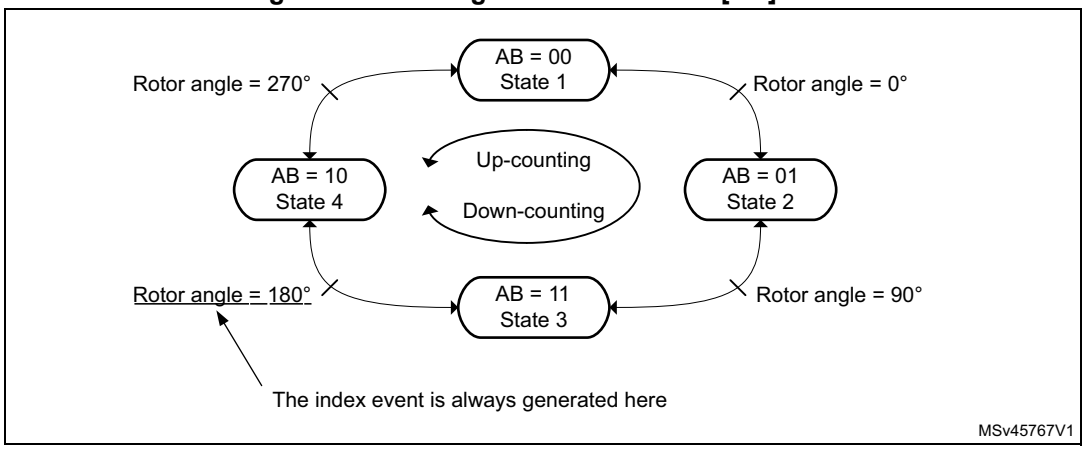


Figure 569 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is '11' (ChA=1, ChB=1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the '11' state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 569. Counter reading with index gated on channel A (IPOS[1:0] = 11)

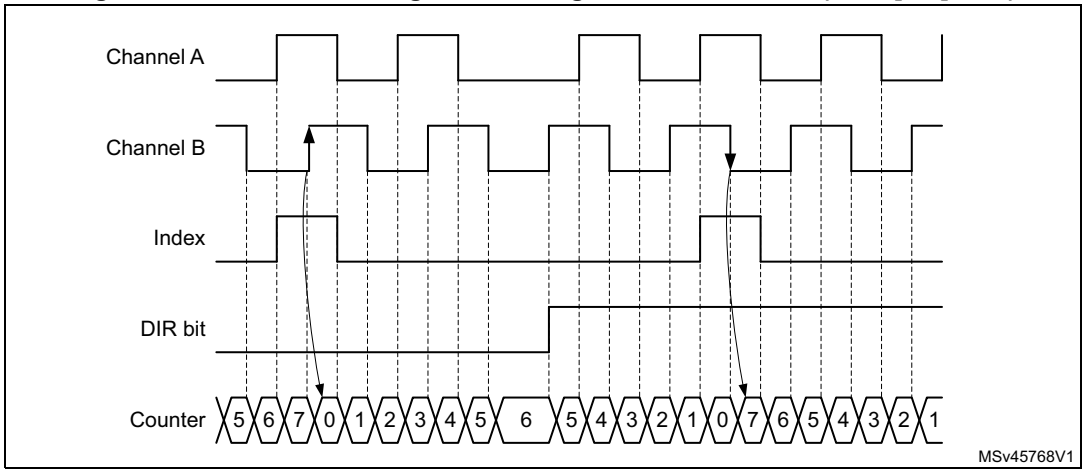


Figure 570 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 570. Counter reading with index ungated (IPOS[1:0] = 00)

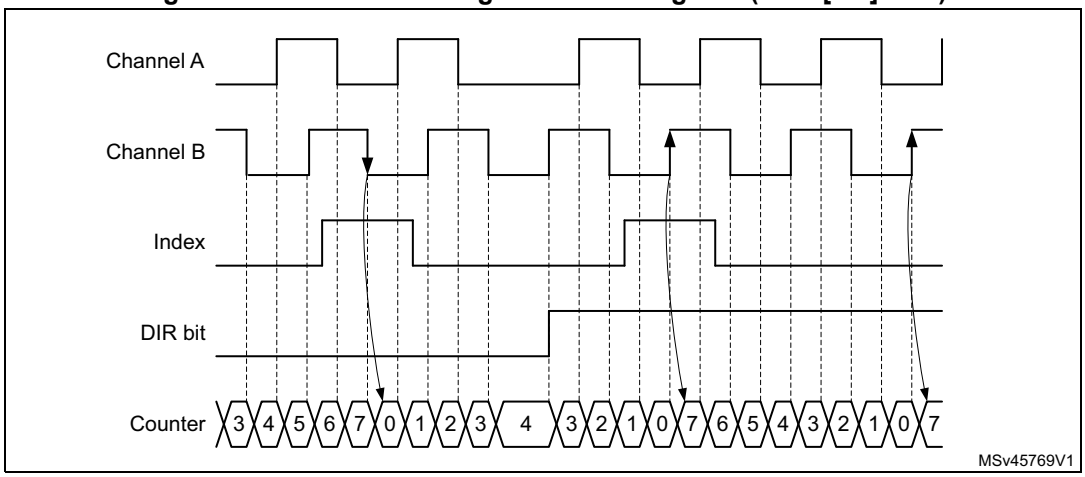


Figure 571 shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 571. Counter reading with index gated on channel A and B

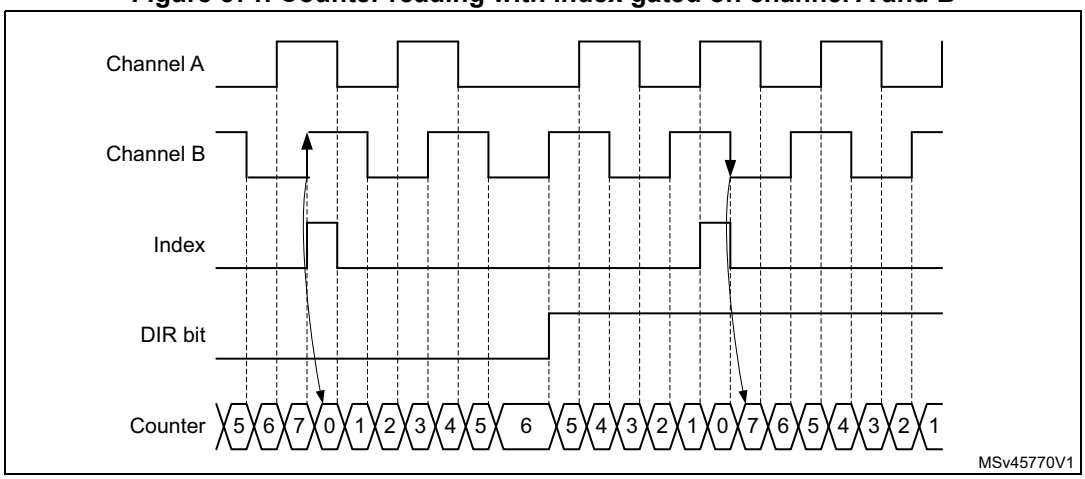


Figure 572 and *Figure 573* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 572. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

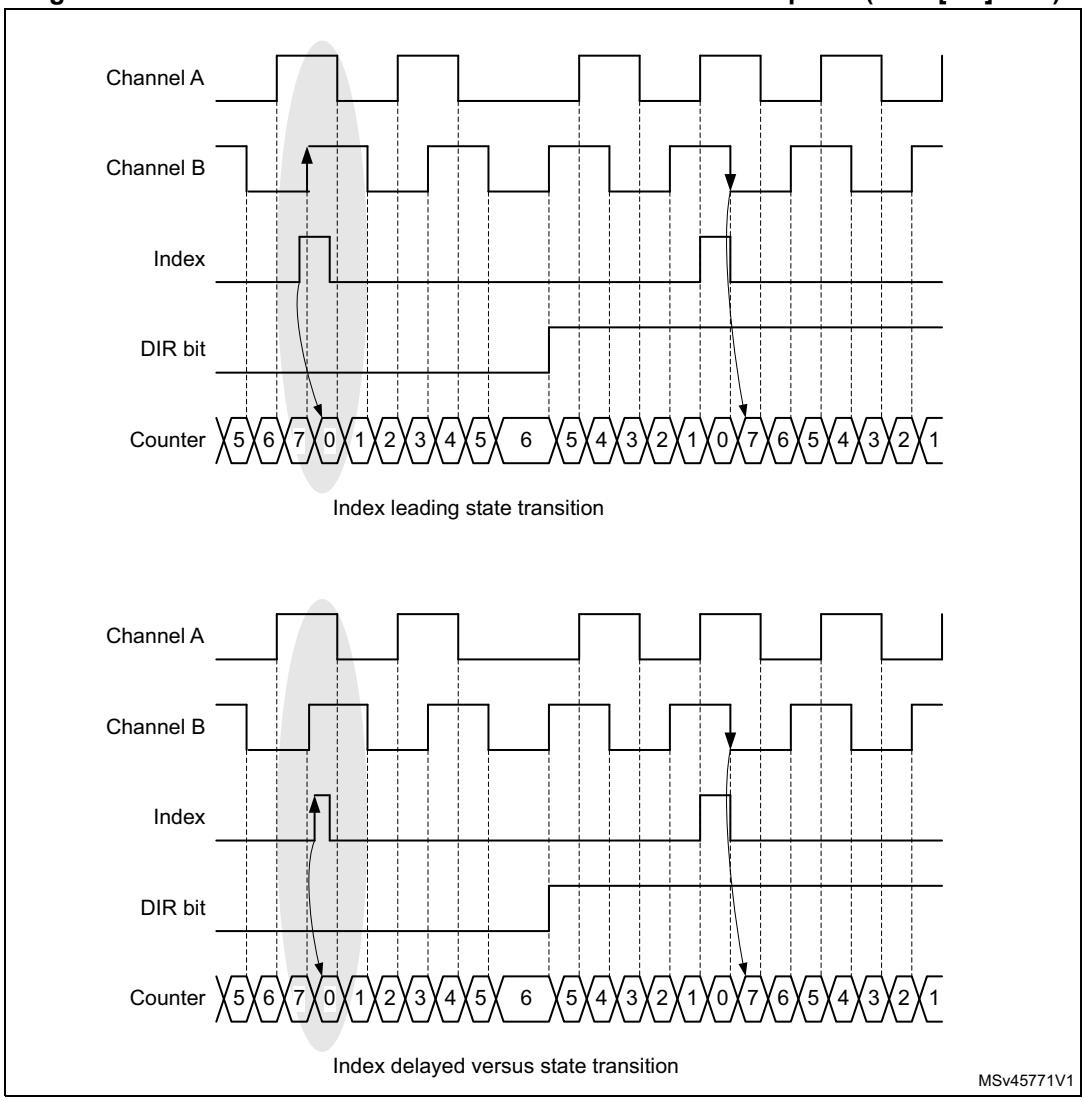


Figure 573. Counter reset Narrow index pulse (closer view, ARR = 0x07)

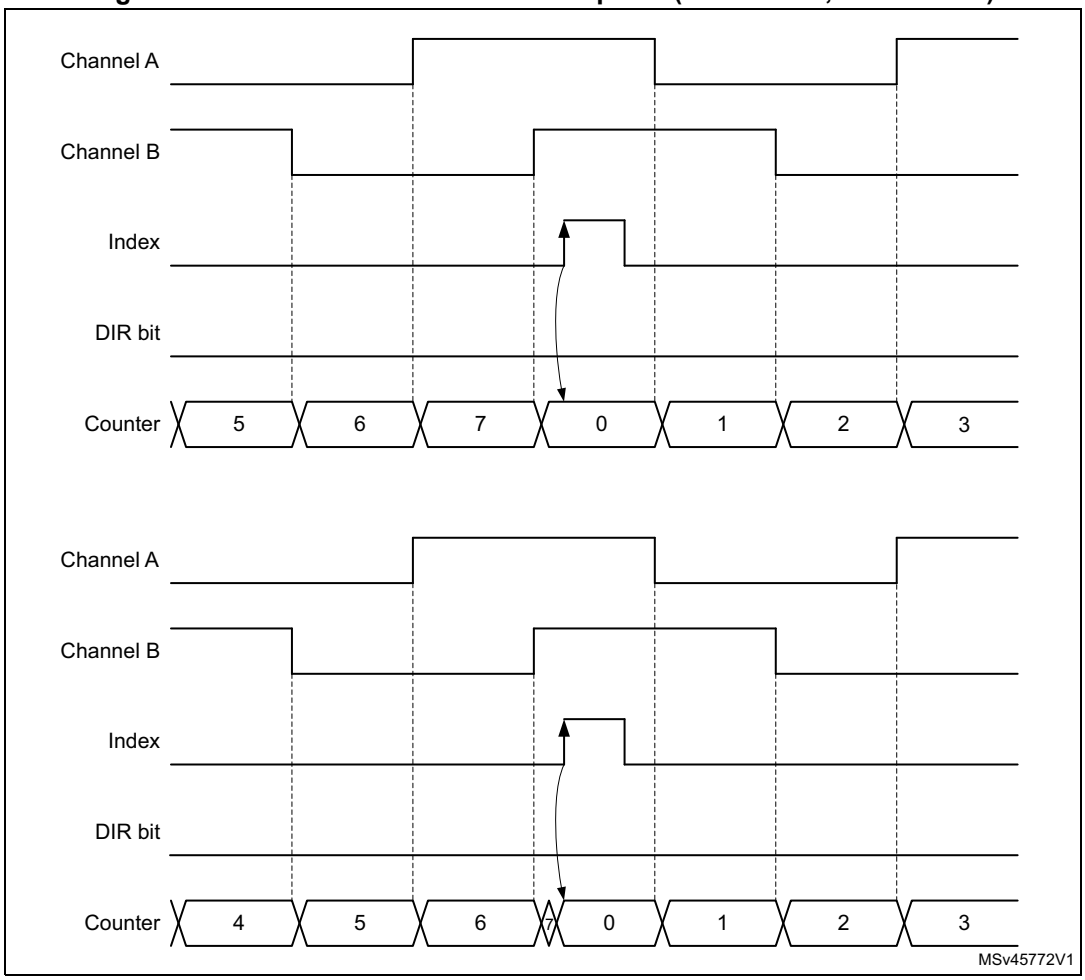
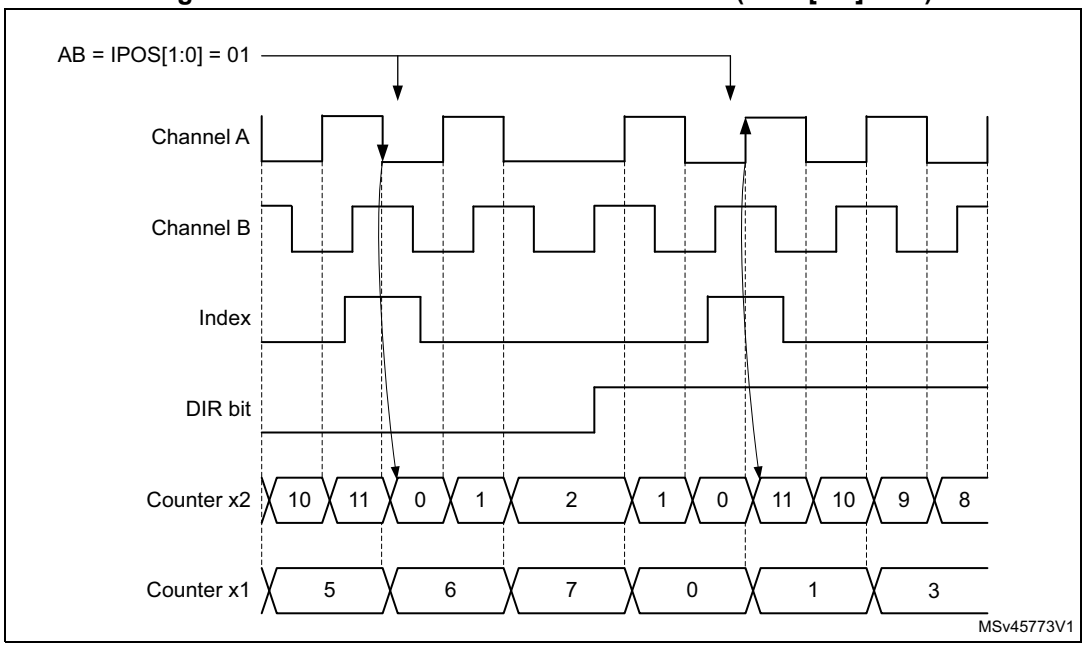


Figure 574 shows how the index is managed in x1 and x2 modes.

Figure 574. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)



Directional index sensitivity

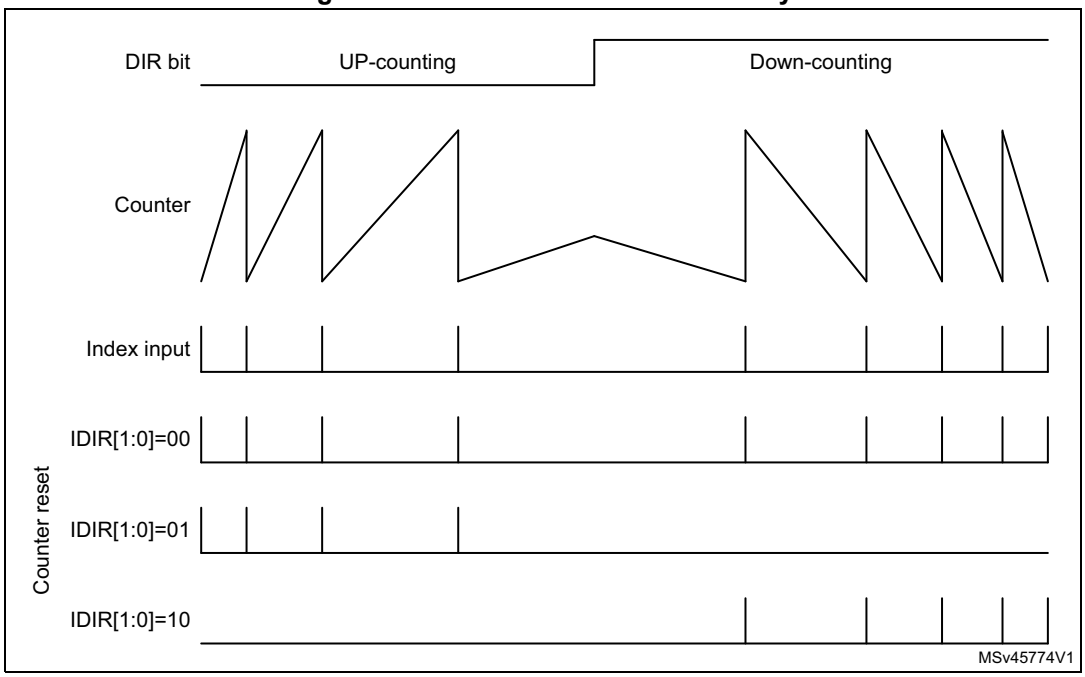
The IDIR[1:0] bit field in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 575 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

Note: The IDR[1:0] bit field must be written when IE bit is reset (index mode disabled).

Note: The directional index sensitivity is not supported in clock + direction mode. When SMS[3:0] = 1010 or 1011, the IDIR[1:0] must be set to 00.

Figure 575. Directional index sensitivity

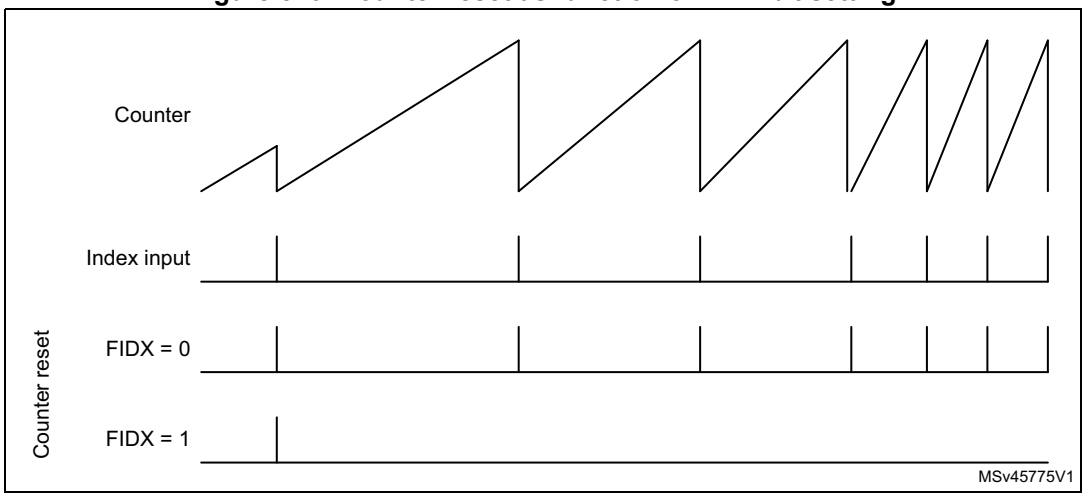


Special first index event management

The FIDX bit in the TIMx_ECR register allows the Index to be taken only once, as shown in *Figure 576*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be re-armed by writing the FIDX bit to 0 and setting it again to 1.

Note: When FIDX = 1, the index can be issued twice (IDXF flag set) if the direction changes at position 0 (index active).

Figure 576. Counter reset as function of FIDX bit setting



Index management in non-quadrature mode

Figure 577 and *Figure 578* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bit field is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as follows:

- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

Figure 577. Index behavior in clock + direction mode, IPOS[0] = 1

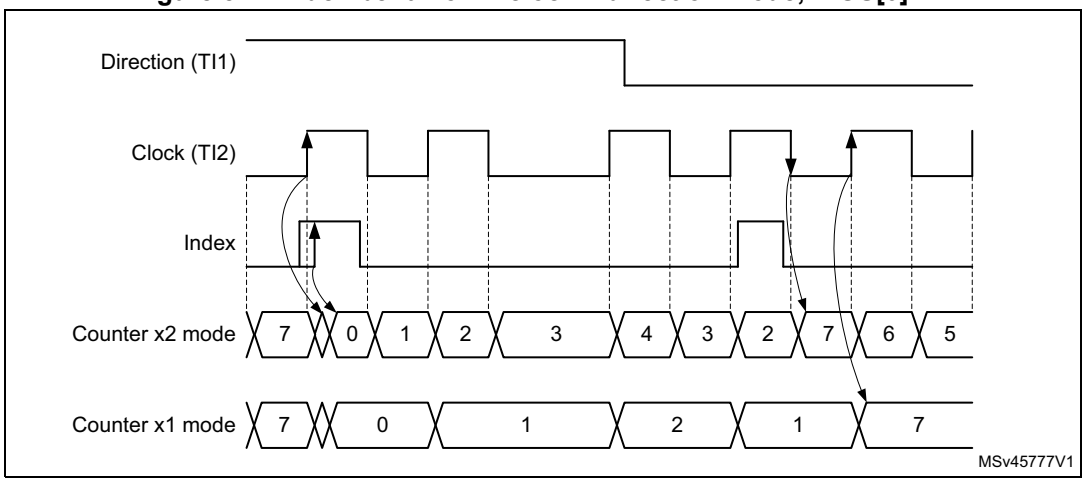
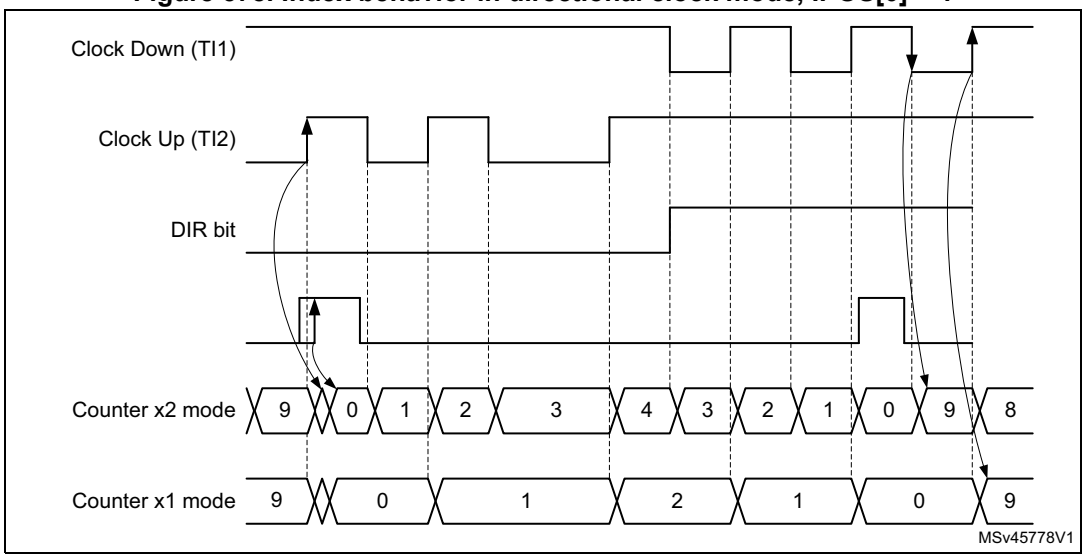


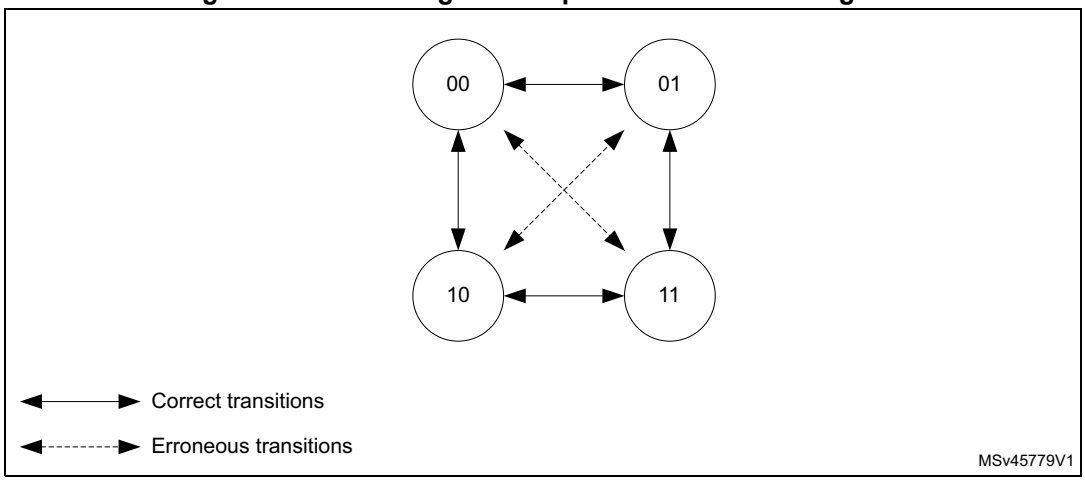
Figure 578. Index behavior in directional clock mode, IPOS[0] = 1



Encoder error management

For encoder configurations where 2 quadrature signals are available, it is possible to detect transition errors. The reading on the 2 inputs corresponds to a 2-bit gray code which can be represented as a state diagram, in *Figure 579.* A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 579. State diagram for quadrature encoded signals



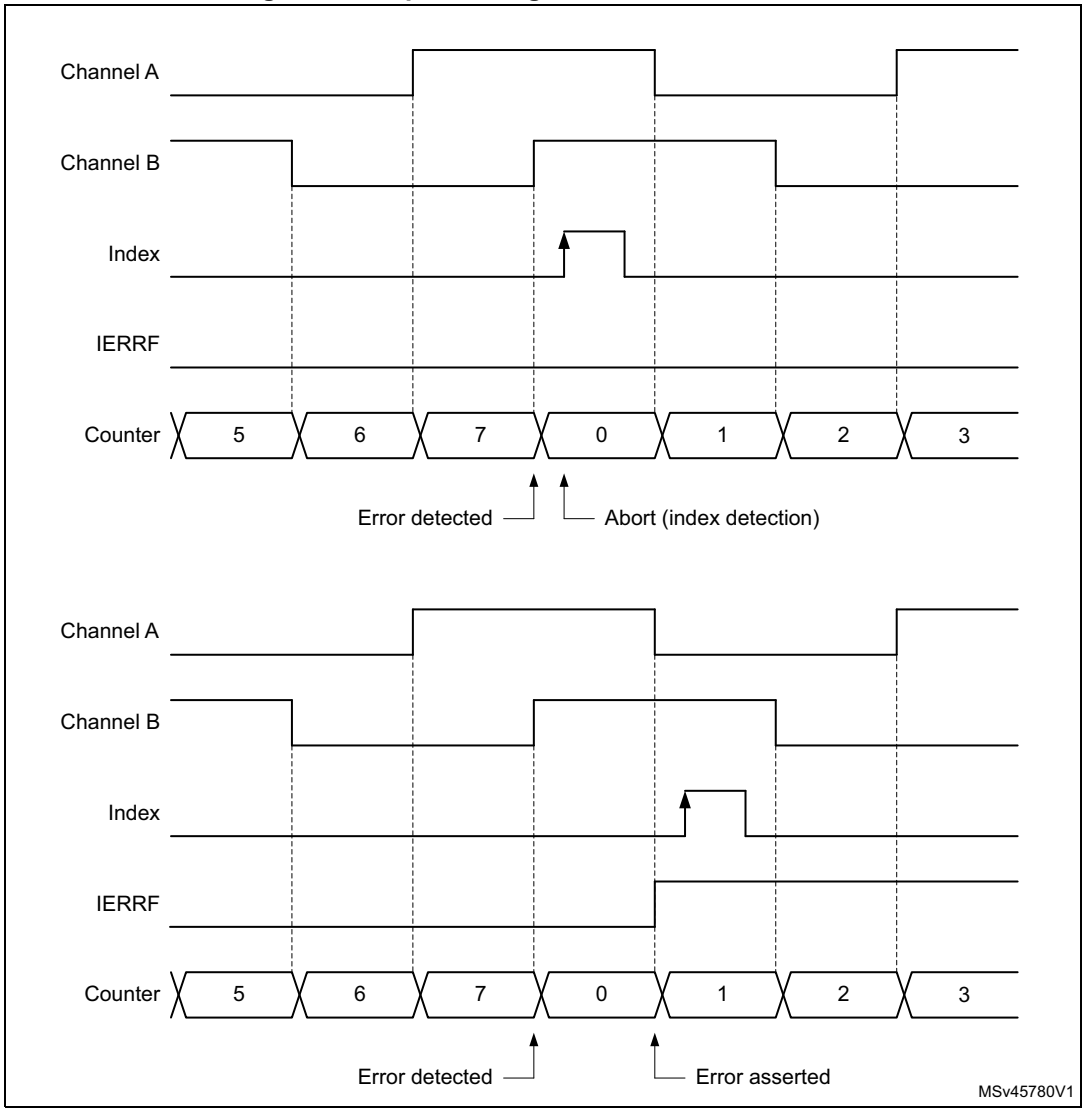
For encoder having an Index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The Index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an Index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

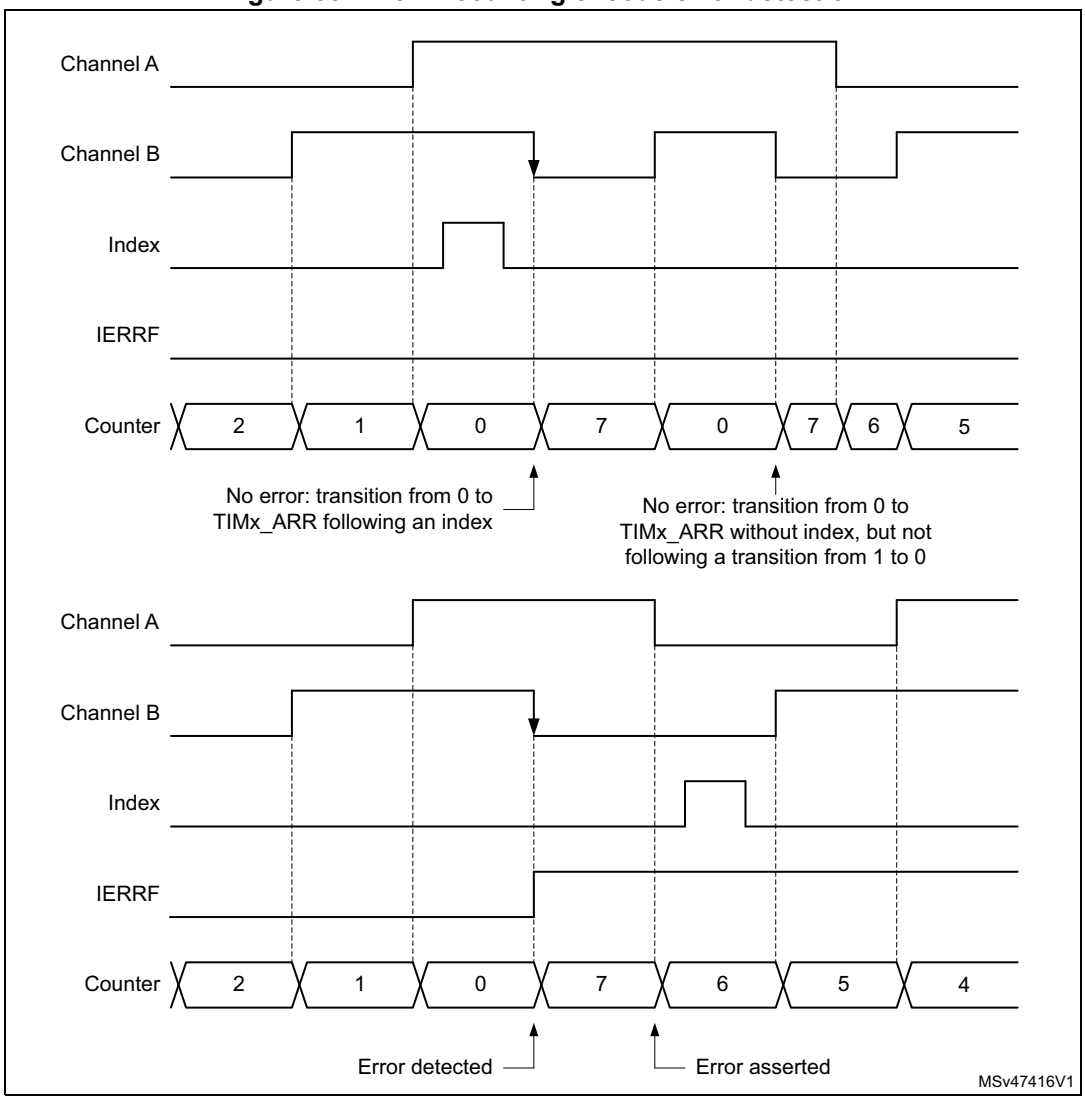
The error assertion is delayed to the transition 0 to 1 when in up-counting. This copes with narrow index pulses in gated A and B mode, as shown in *Figure 580*.

Figure 580. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown in *Figure 581*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 581. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder Interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the Index event sets the IDXF interrupt flag in the TIMx_SR status register. An Index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

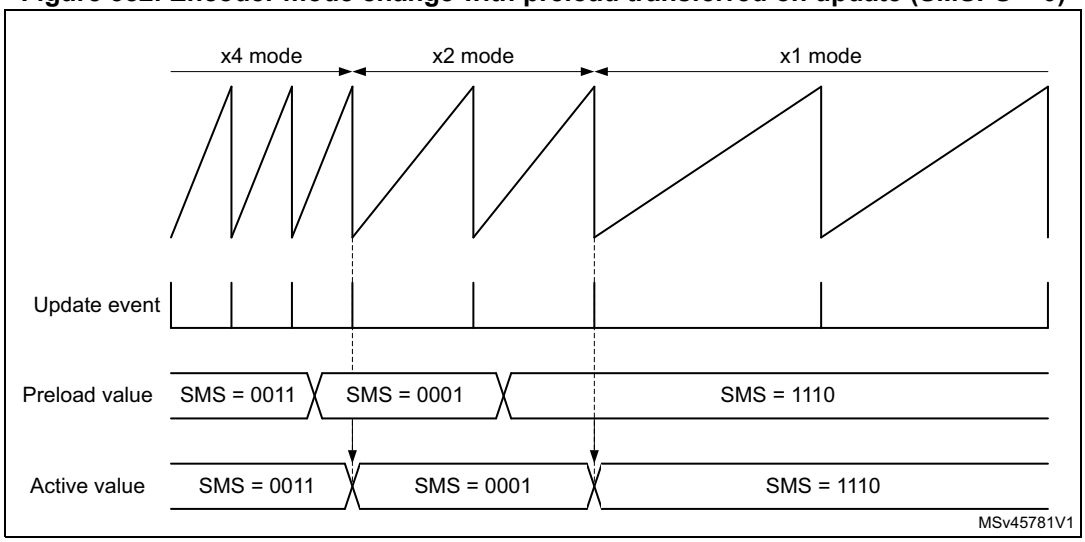
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the Update interrupt rate, by switching from x4 to x2 to x1 mode, as shown in *Figure 582*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting. This mode must be used only when index is disabled (bit IE = 0).
- SMSPS = 1: the transfer is triggered by the Index event.

Figure 582. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bit field to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

36.4.19 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bit field to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

36.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

36.4.21 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3 (the output of the XOR gate combines the two input pins tim_ti1 and tim_ti2 only on TIM5).

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 35.3.29: Interfacing with Hall sensors*.

36.4.22 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

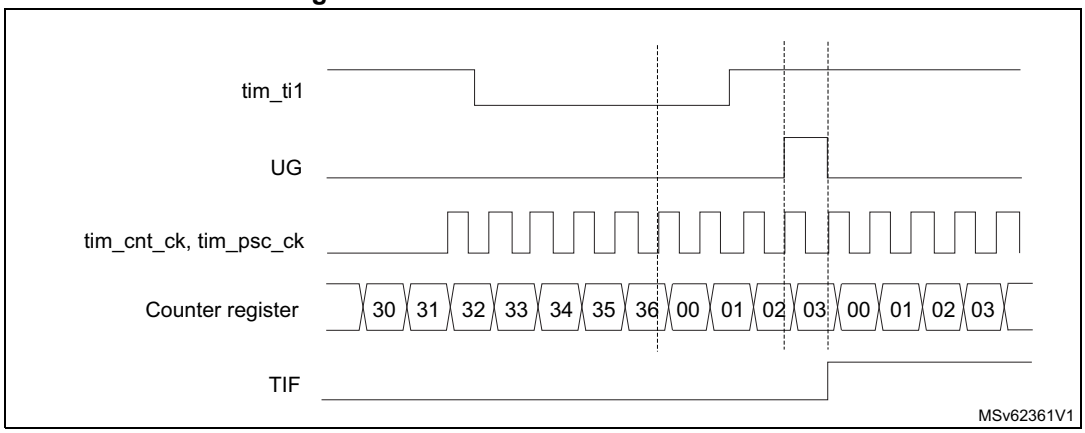
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 583. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

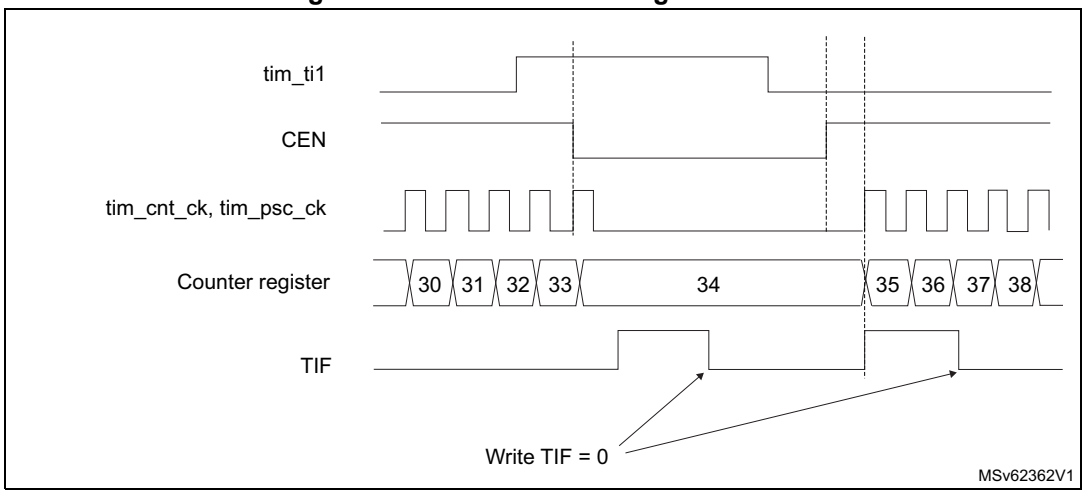
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 584. Control circuit in gated mode



Note: The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

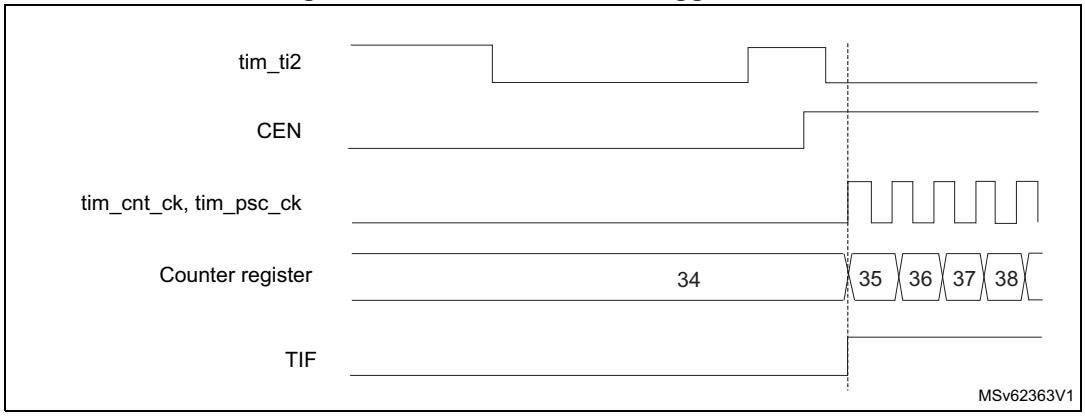
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 585. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode allows to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

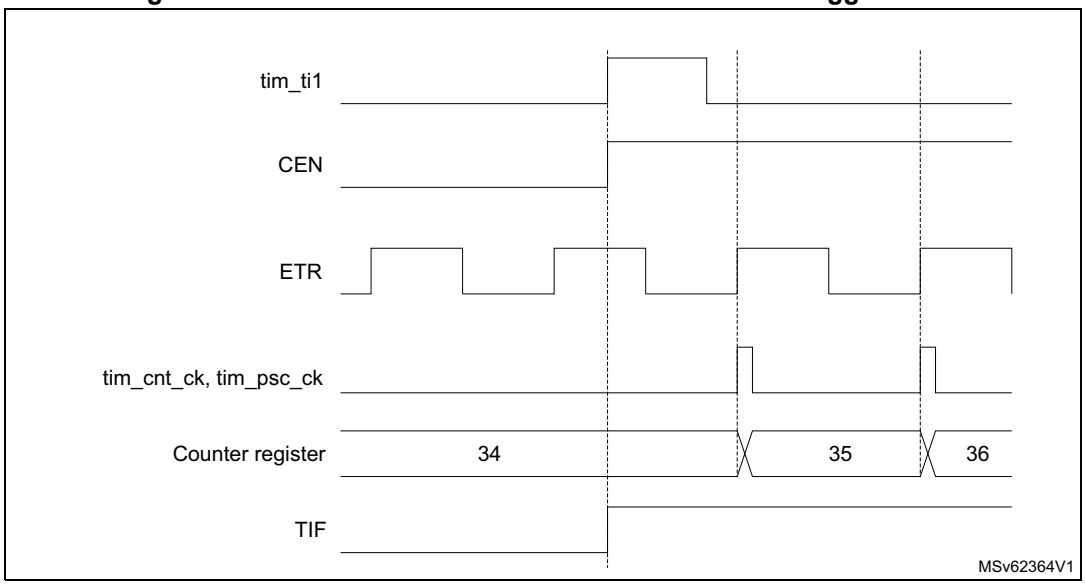
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on tim_etr_in and ECE=1 to enable the external clock mode 2
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter
   - The capture prescaler is not used for triggering and does not need to be configured
   - CC1S=01in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only)
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 586. Control circuit in external clock mode 2 + trigger mode



36.4.23 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 587 and *Figure 588* show examples of master/slave timer connections.

Figure 587. Master/Slave timer example

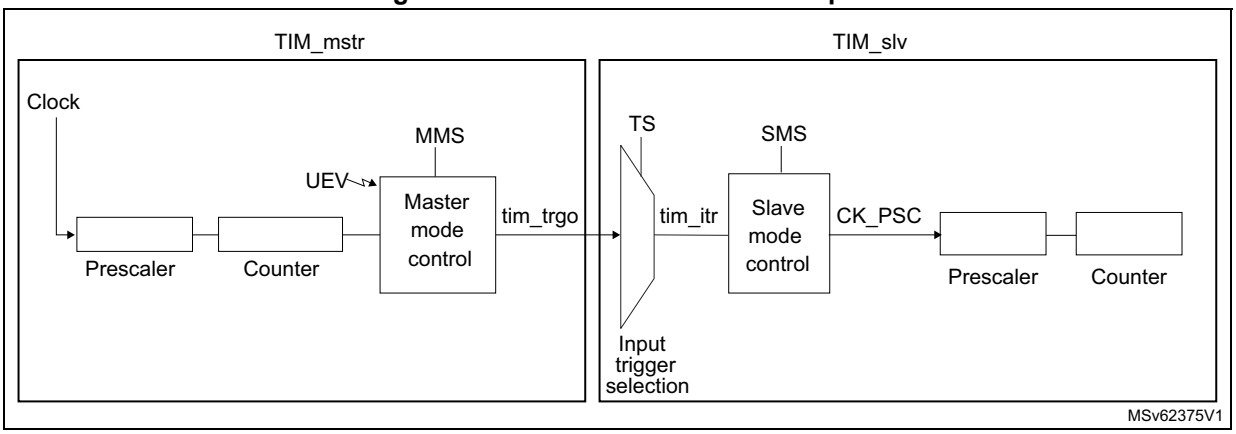
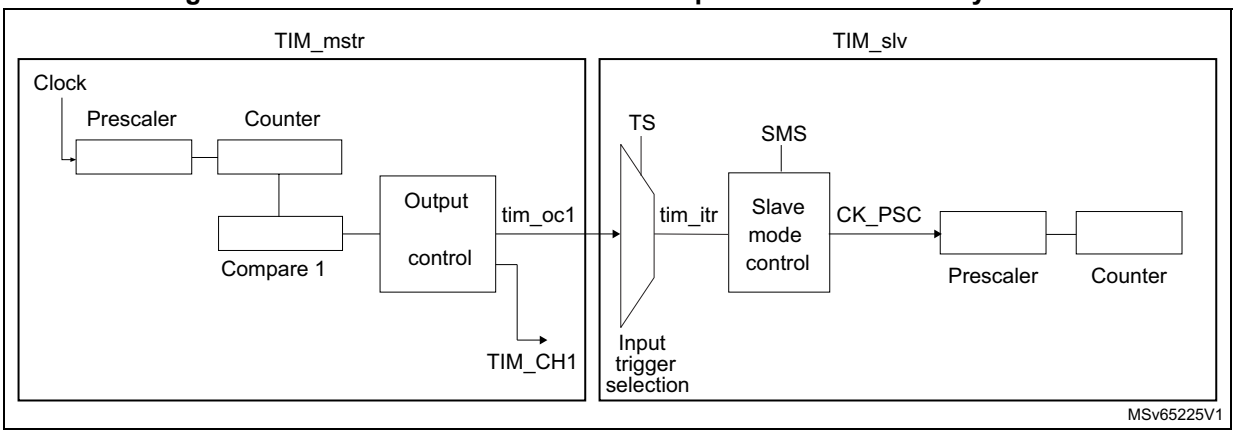


Figure 588. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (refer to Figure 588) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 36.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals). The tim_oc1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.
For instance, if the destination timer tim_ker_ck clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 587*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS=00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which corresponds to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM_slv.

Using one timer to enable another timer

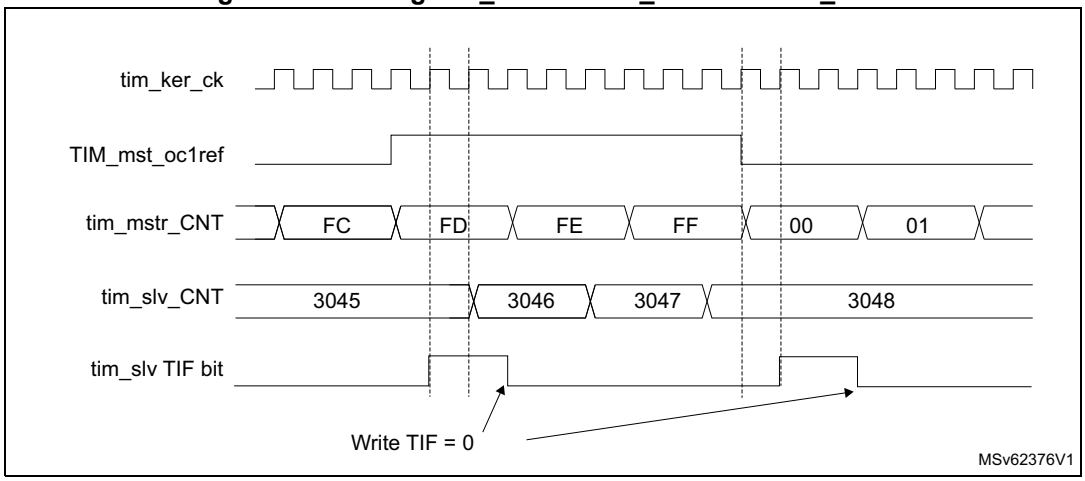
In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 587* for connections. TIM_slv counts on the divided internal clock only when

tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

1. Configure TIM_mstr master mode to send its Output Compare 1 Reference (tim_oc1ref) signal as trigger output (MMS=100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS=101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing ‘1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing ‘1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 589. Gating TIM_slv with tim_oc1ref of TIM_mstr



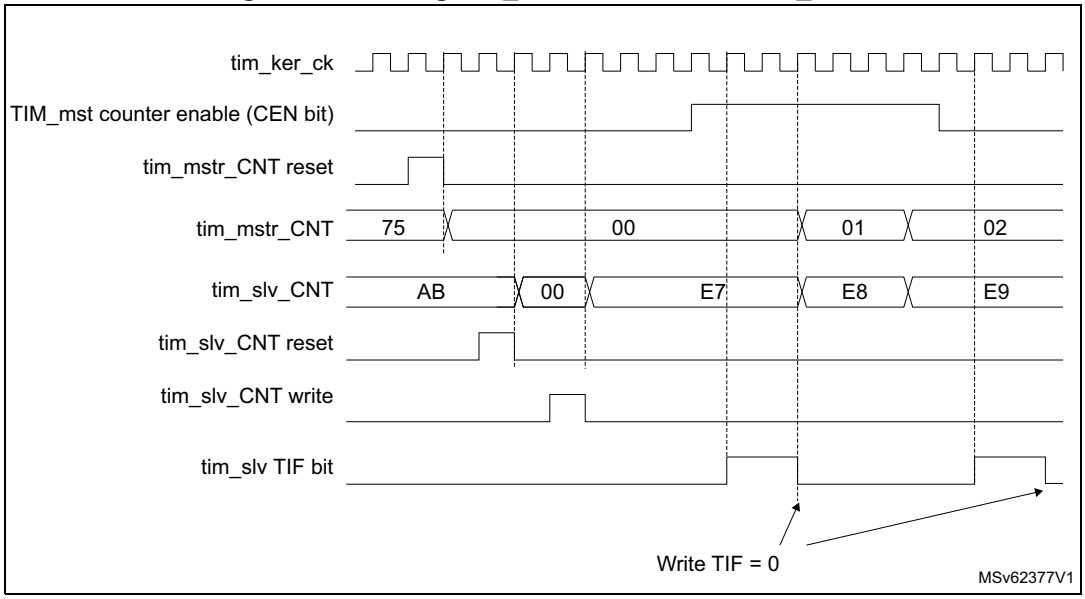
In the example in *Figure 589*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 590*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing ‘0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its Output Compare 1 Reference (tim_oc1ref) signal as trigger output (MMS=100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS=101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing ‘1 in UG bit (TIM_mstr_EGR register).

6. Reset TIM_slv by writing ‘1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing ‘0xE7’ in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing ‘1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing ‘1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing ‘0 in the CEN bit (TIM_mstr_CR1 register).

Figure 590. Gating TIM_slv with Enable of TIM_mstr

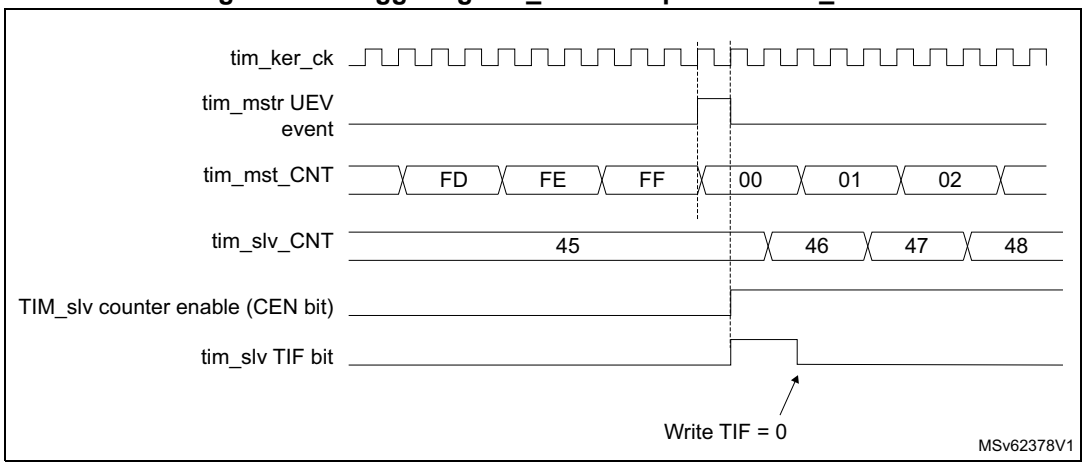


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 587* for connections. TIM_slv starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

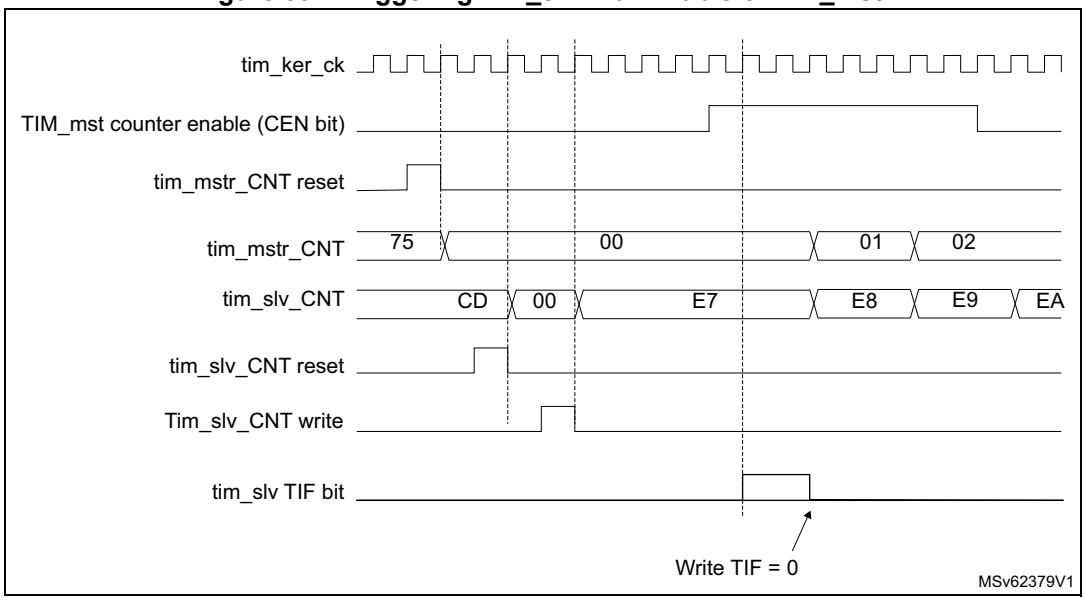
1. Configure TIM_mstr master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS=110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing ‘1 in the CEN bit (TIM_mstr_CR1 register).

Figure 591. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 592* shows the behavior with the same configuration as in *Figure 591* but in trigger mode instead of gated mode (SMS=110 in the TIM_slv_SMCR register).

Figure 592. Triggering TIM_slv with Enable of TIM_mstr



Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 587* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

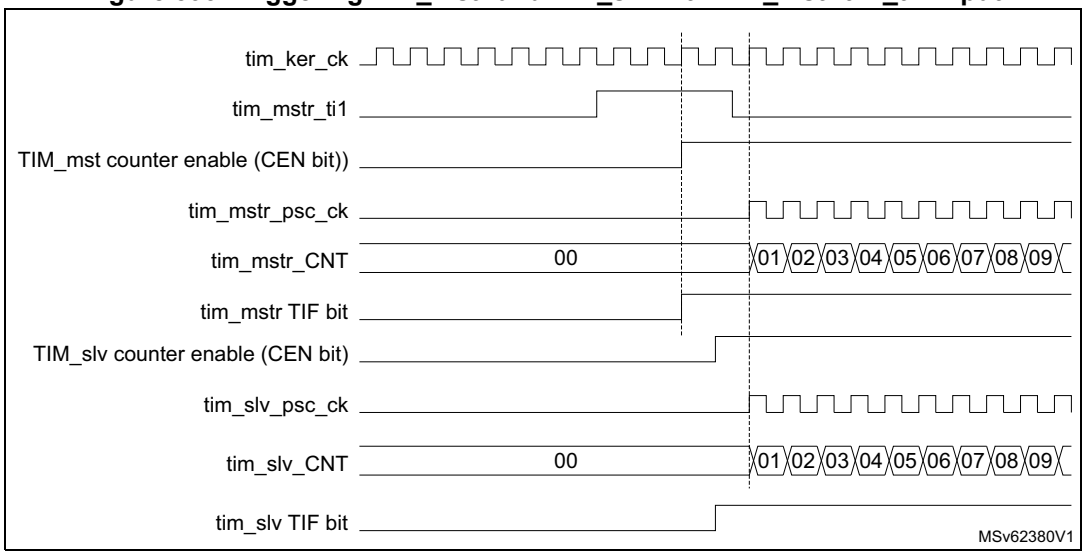
1. Configure TIM_mstr master mode to send its Enable as trigger output (MMS=001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS=00100 in the TIM_mstr_SMCR register).

3. Configure TIM_mstr in trigger mode (SMS=110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM=1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS=110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters start from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 593. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (timer, ADC...) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

36.4.24 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), that is the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (refer to the note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

36.4.25 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in *Table 431*.

Table 431. DMA request

DMA acronym	DMA request	Enable control bit
TIM_UP	Update	UDE
TIM_CH1	Capture/compare 1	CC1DE
TIM_CH2	Capture/compare 2	CC2DE
TIM_CH3	Capture/compare 3	CC3DE

Table 431. DMA request (continued)

DMA acronym	DMA request	Enable control bit
TIM_CH4	Capture/compare 4	CC4DE
TIM_TRIG	Trigger	TDE

36.4.26 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

36.4.27 TIM2/TIM3/TIM4/TIM5 low-power modes

Table 432. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.

36.4.28 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in *Table 433*.

Table 433. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
TIM_UP	Update	UIF	UIE	Write 0 in UIF	Yes
TIM_CC	Capture/compare 1	CC1IF	CC1IE	Write 0 in CC1IF	Yes
	Capture/compare 2	CC2IF	CC2IE	Write 0 in CC2IF	Yes
	Capture/compare 3	CC3IF	CC3IE	Write 0 in CC3IF	Yes
	Capture/compare 4	CC4IF	CC4IE	Write 0 in CC4IF	Yes
TIM_TRG	Trigger	TIF	TIE	Write 0 in TIF	Yes
TIM_DIR_IDX	Index	IDXF	IDXIE	Write 0 in IDXF	Yes
	Direction	DIRF	DIRIE	Write 0 in DIRF	Yes
TIM_IERR	Index Error	IERRF	IERRIE	Write 0 in IERRF	Yes
TIM_TER	Transition Error	TERRF	TERRIE	Write 0 in TERRF	Yes

36.5 TIM2/TIM3/TIM4/TIM5 registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

36.5.1 TIM2/TIM3/TIM4/TIM5 memory map

Table 434. TIM2/TIM3/TIM4/TIM5 register memory map

Offset	Register name
0x000	*TIMx control register 1 (TIMx_CR1)(x = 2 to 5)*
0x004	*TIMx control register 2 (TIMx_CR2)(x = 2 to 5)*
0x008	*TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)*
0x00C	*TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)*
0x010	*TIMx status register (TIMx_SR)(x = 2 to 5)*
0x014	*TIMx event generation register (TIMx_EGR)(x = 2 to 5)*
0x018	*TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)*
0x018	*TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)*
0x01C	*TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)*
0x01C	*TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)*
0x020	*TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)*
0x024	*TIMx counter (TIMx_CNT)(x = 3, 4)*
0x024	*TIMx counter (TIMx_CNT)(x = 2, 5)*
0x028	*TIMx prescaler (TIMx_PSC)(x = 2 to 5)*
0x02C	*TIMx auto-reload register (TIMx_ARR)(x = 3, 4)*
0x02C	*TIMx auto-reload register (TIMx_ARR)(x = 2, 5)*
0x034	*TIMx capture/compare register 1 (TIMx_CCR1)(x = 3, 4)*
0x034	*TIMx capture/compare register 1 (TIMx_CCR1)(x = 2, 5)*
0x038	*TIMx capture/compare register 2 (TIMx_CCR2)(x = 3, 4)*
0x038	*TIMx capture/compare register 2 (TIMx_CCR2)(x = 2, 5)*
0x03C	*TIMx capture/compare register 3 (TIMx_CCR3)(x = 3, 4)*
0x03C	*TIMx capture/compare register 3 (TIMx_CCR3)(x = 2, 5)*
0x040	*TIMx capture/compare register 4 (TIMx_CCR4)(x = 3, 4)*
0x040	*TIMx capture/compare register 4 (TIMx_CCR4)(x = 2, 5)*
0x058	*TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5)*
0x05C	*TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)*

Table 434. TIM2/TIM3/TIM4/TIM5 register memory map (continued)

Offset	Register name
0x060	*TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5)*
0x064	*TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5)*
0x3DC	*TIMx DMA control register (TIMx_DCR)(x = 2 to 5)*
0x3E0	*TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)*

36.5.2 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit field indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

36.5.3 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15..0] multiplexer output is to tim_ti1 input
1: The tim_ti1_in[15..0], tim_ti2_in[15..0] and tim_ti3_in[15..0] multiplexers outputs are XORed and connected to the tim_ti1 input. Refer also to *Section 35.3.29: Interfacing with Hall sensors*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

36.5.4 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source
This bit selects whether the events that triggers the SMS[3:0] bit field transfer from preload to active
0: The transfer is triggered by the Timer's Update event
1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable
This bit selects whether the SMS[3:0] bit field is preloaded
0: SMS[3:0] bit field is not preloaded
1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity
This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations
0: tim_etr_in is non-inverted, active at high level or rising edge
1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

*Note: **1**: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS=111 and TS=00111).*
***2**: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).*
***3**: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etrp frequency divided by 2
10: tim_etrp frequency divided by 4
11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit field then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection (see bits 21:20 for TS[4:3])

This bit field selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
Others: Reserved

Refer to *Section 36.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example, when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source

0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 36.3: TIM2/TIM3/TIM4/TIM5 implementation.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC...) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

36.5.5 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

36.5.6 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
								rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to ‘0’.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to ‘0’.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to ‘0’.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to ‘0’.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description) and in retriggerable one pulse mode. It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT has matched the content of the TIMx_CCR1 register.

Note: When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode).

If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred.

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on tim_ic1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

At overflow or underflow and if UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

36.5.7 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

36.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2

0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4

0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit field defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

36.5.9 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **OC1M[3]**: Output Compare 1 mode - bit 3

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Bits 6:4 **OC1M[2:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref=0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved.

1011: Reserved.

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: In PWM mode, the tim_ocref_clr level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

36.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

36.5.11 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

36.5.12 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*
CC1 channel configured as output: CC1NP must be kept cleared in this case.
CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*
CC1 channel configured as output:
0: tim_oc1 active high
1: tim_oc1 active low
CC1 channel configured as input: CC1NP/CC1P bits select tim_ti1fp1 and tim_ti2fp1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to tim_tixfp1 rising edge (capture, trigger in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger in gated mode, encoder mode).
01: inverted/falling edge
Circuit is sensitive to tim_tixfp1 falling edge (capture, trigger in reset, external clock or trigger mode), tim_tixfp1 is inverted (trigger in gated mode, encoder mode).
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both tim_tixfp1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*
CC1 channel configured as output:
0: Off - tim_oc1 is not active
1: On - tim_oc1 signal is output on the corresponding output pin
CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 435. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE=0.

36.5.13 TIMx counter (TIMx_CNT)(x = 3, 4)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
Reserved
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value‘
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[15:0]. The fractional part is not available.

36.5.14 TIMx counter (TIMx_CNT)(x = 2, 5)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT [31] or UIF CPY	CNT[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CNT[31] or UIFCPY**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

36.5.15 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

36.5.16 TIMx auto-reload register (TIMx_ARR)(x = 3, 4)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Low Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to *Section 36.4.3: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bit field contains the dithered part.

36.5.17 TIMx auto-reload register (TIMx_ARR)(x = 2, 5)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARR[31:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to *Section 36.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bit field contains the dithered part.

36.5.18 TIMx capture/compare register 1 (TIMx_CCR1)(x = 3, 4)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bit field contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR1[15:0] bits hold the capture value. The CCR1[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:0]. The CCR1[3:0] bits are reset.

36.5.19 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2, 5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bit field contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

36.5.20 TIMx capture/compare register 2 (TIMx_CCR2)(x = 3, 4)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 1 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bit field contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR2[15:0] bits hold the capture value. The CCR2[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:0]. The CCR2[3:0] bits are reset.

36.5.21 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2, 5)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bit field contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

36.5.22 TIMx capture/compare register 3 (TIMx_CCR3)(x = 3, 4)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bit field contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR3[15:0] bits hold the capture value. The CCR3[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[19:0]. The CCR3[3:0] bits are reset.

36.5.23 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2, 5)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bit field contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

36.5.24 TIMx capture/compare register 4 (TIMx_CCR4)(x = 3, 4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bit field contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR4[15:0] bits hold the capture value. The CCR4[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:0]. The CCR4[3:0] bits are reset.

36.5.25 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2, 5)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bit field contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

36.5.26 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	Res.	Res.	IDIR[1:0]		IE
								rw	rw	rw			rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bit field sets the clock prescaler for the pulse generator, as follows:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bit field defines the pulse duration, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicate on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Note: The IDR[1:0] bit field must be written when IE bit is reset (index disabled).

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

36.5.27 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[0..15] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[0..15] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[0..15] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[0..15] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

36.5.28 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

36.5.29 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:15 **OCRSEL[2:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
...
111: tim_ocref_clr7
Refer to *Section 36.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

36.5.30 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), that is the number of transfers. Transfers can be in half-words or in bytes (see the example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer must be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

36.5.31 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

37 Basic timers (TIM6/TIM7)

37.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist in a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

37.2 TIM6/TIM7 main features

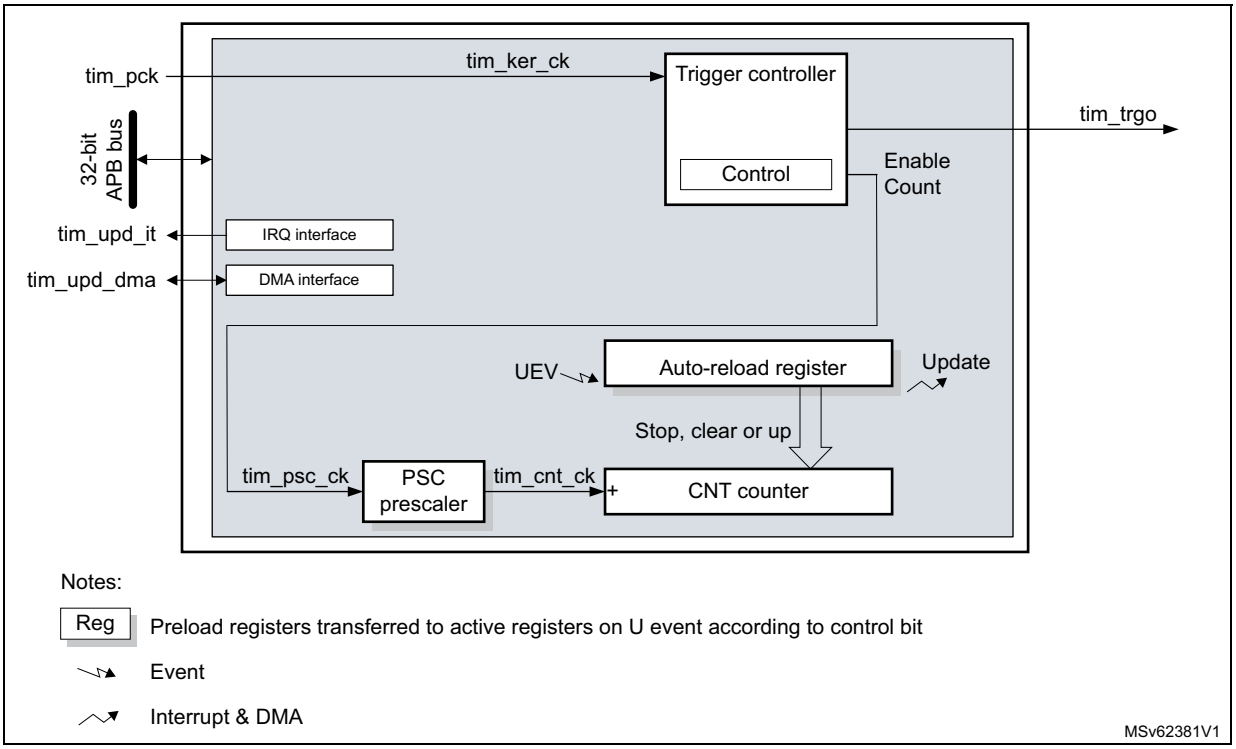
Basic timer (TIM6/TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

37.3 TIM6/TIM7 functional description

37.3.1 TIM6/TIM7 block diagram

Figure 594. Basic timer block diagram



37.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 436. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

37.3.3 TIM6/TIM7 clocks

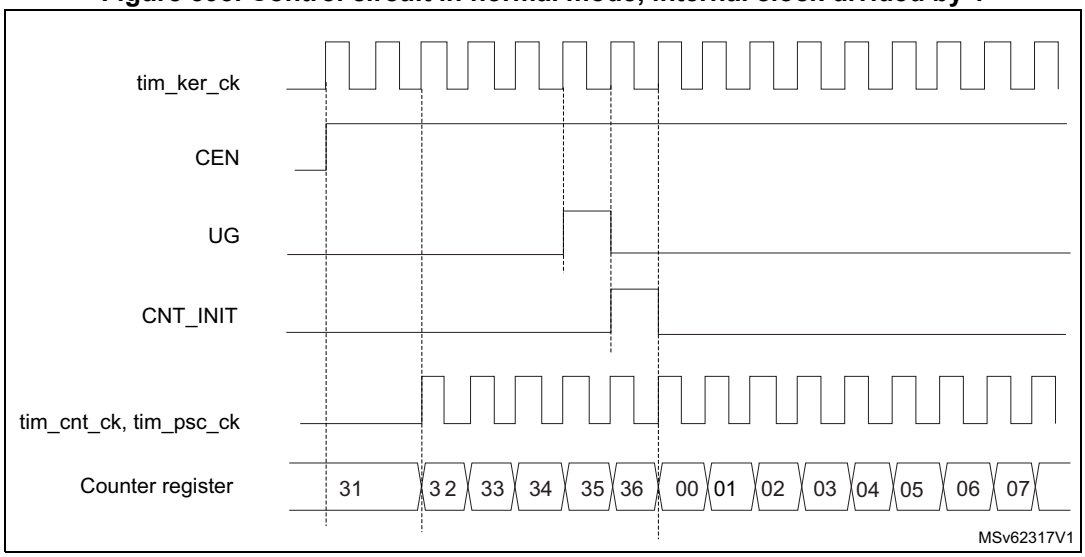
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 595 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 595. Control circuit in normal mode, internal clock divided by 1



37.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set 1 clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 596 and *Figure 597* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 596. Counter timing diagram with prescaler division change from 1 to 2

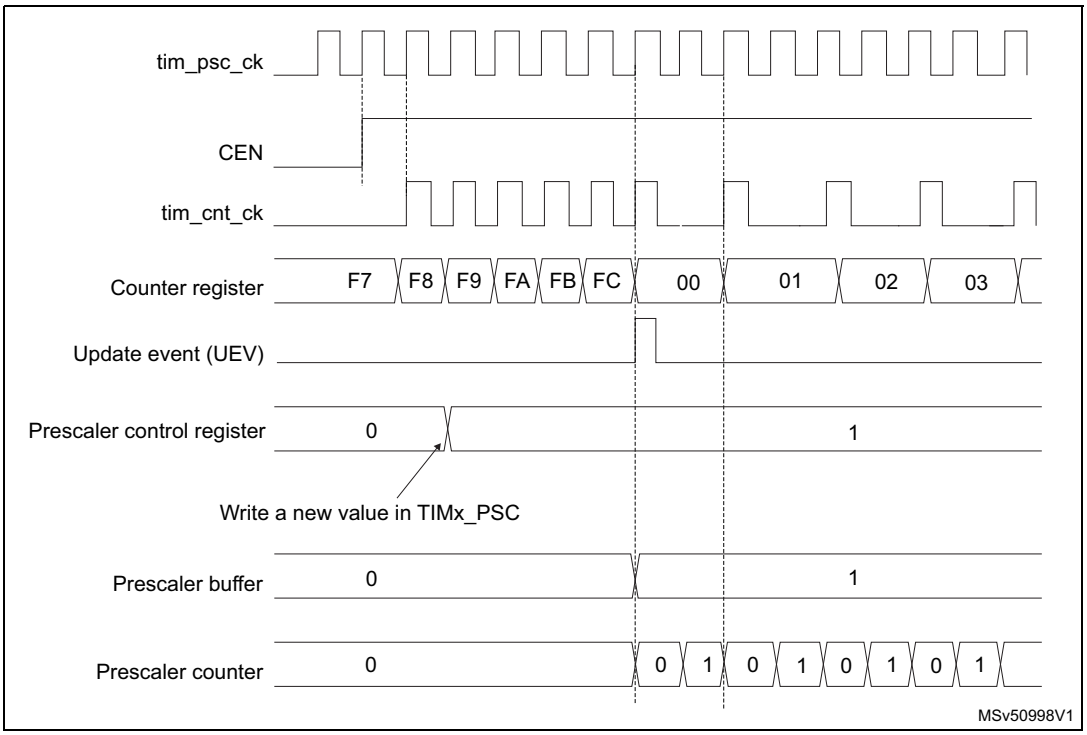
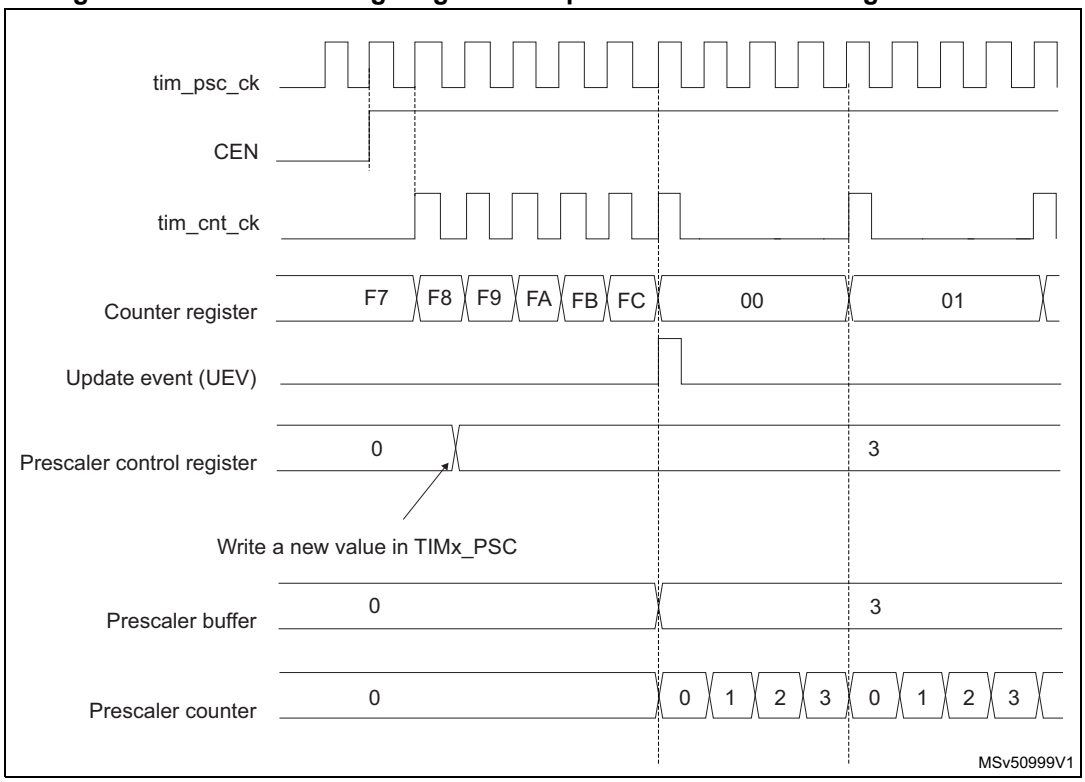


Figure 597. Counter timing diagram with prescaler division change from 1 to 4



37.3.5 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 598. Counter timing diagram, internal clock divided by 1

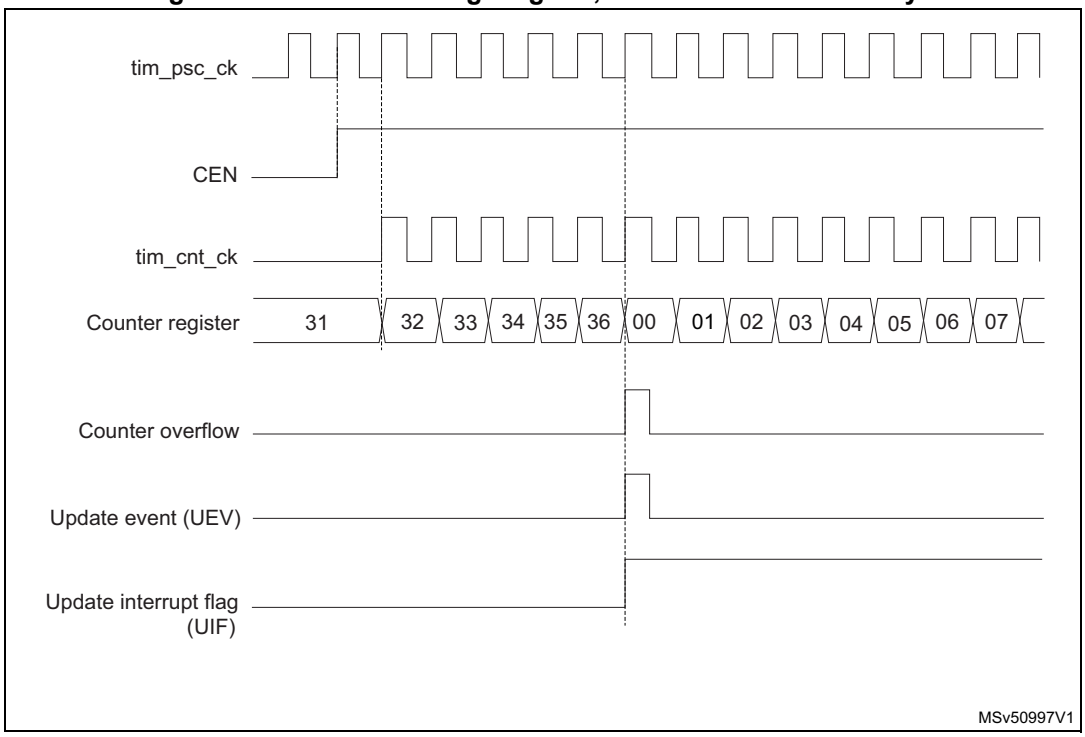


Figure 599. Counter timing diagram, internal clock divided by 2

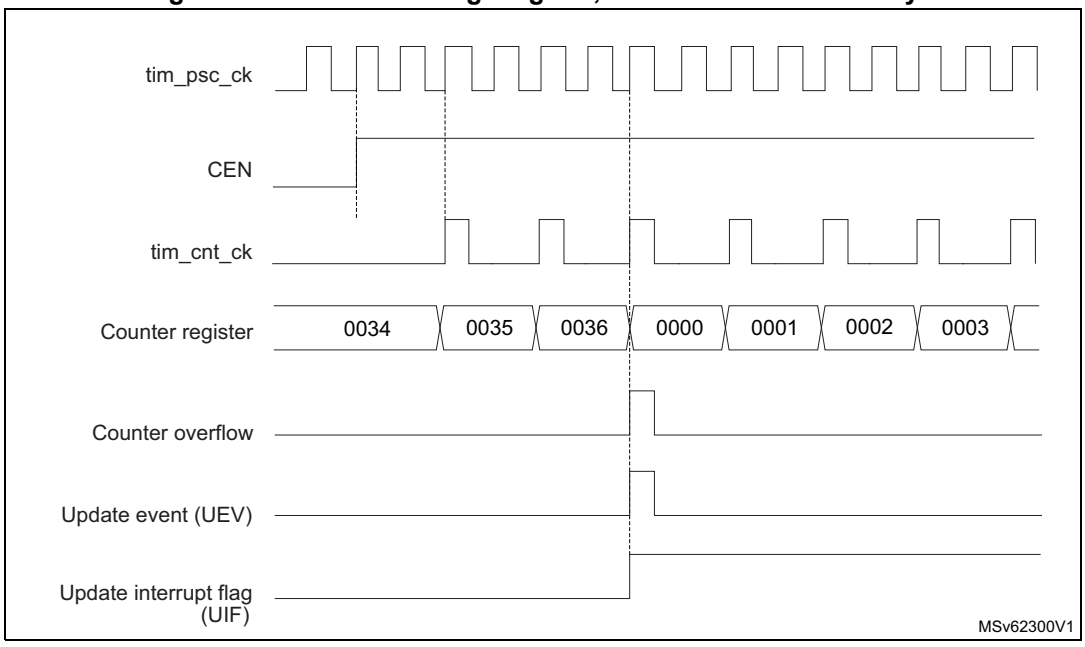


Figure 600. Counter timing diagram, internal clock divided by 4

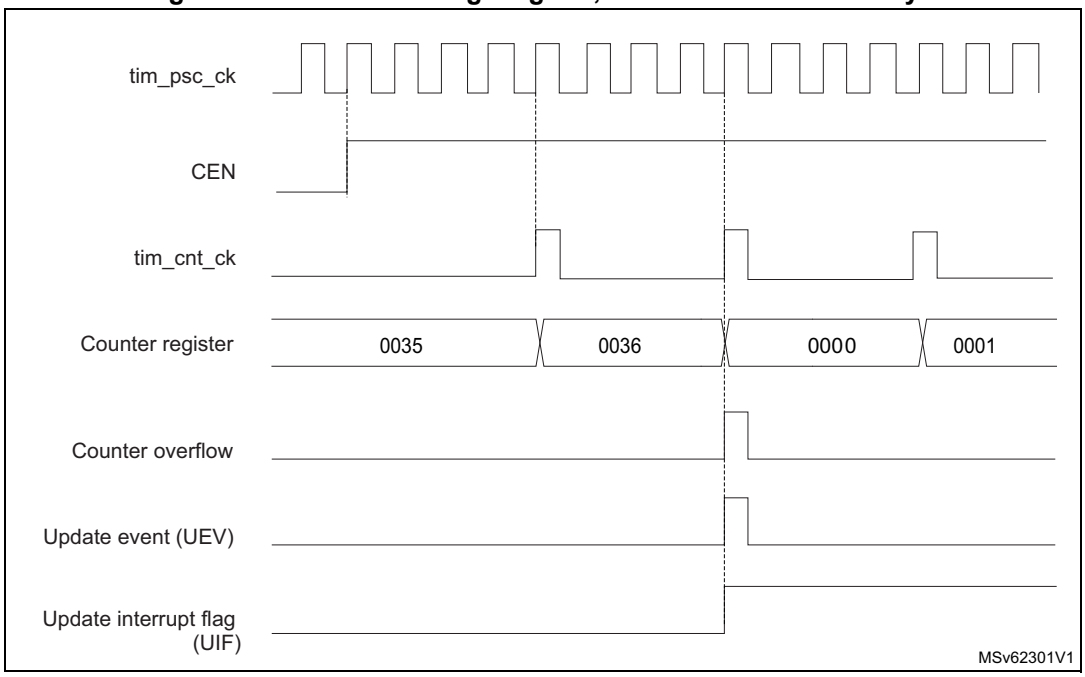


Figure 601. Counter timing diagram, internal clock divided by N

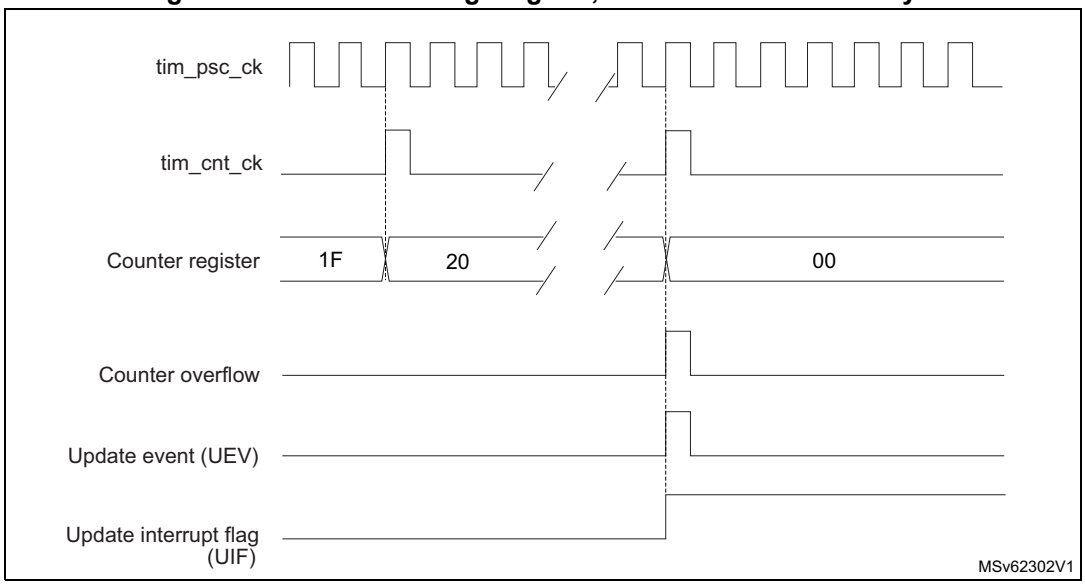


Figure 602. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

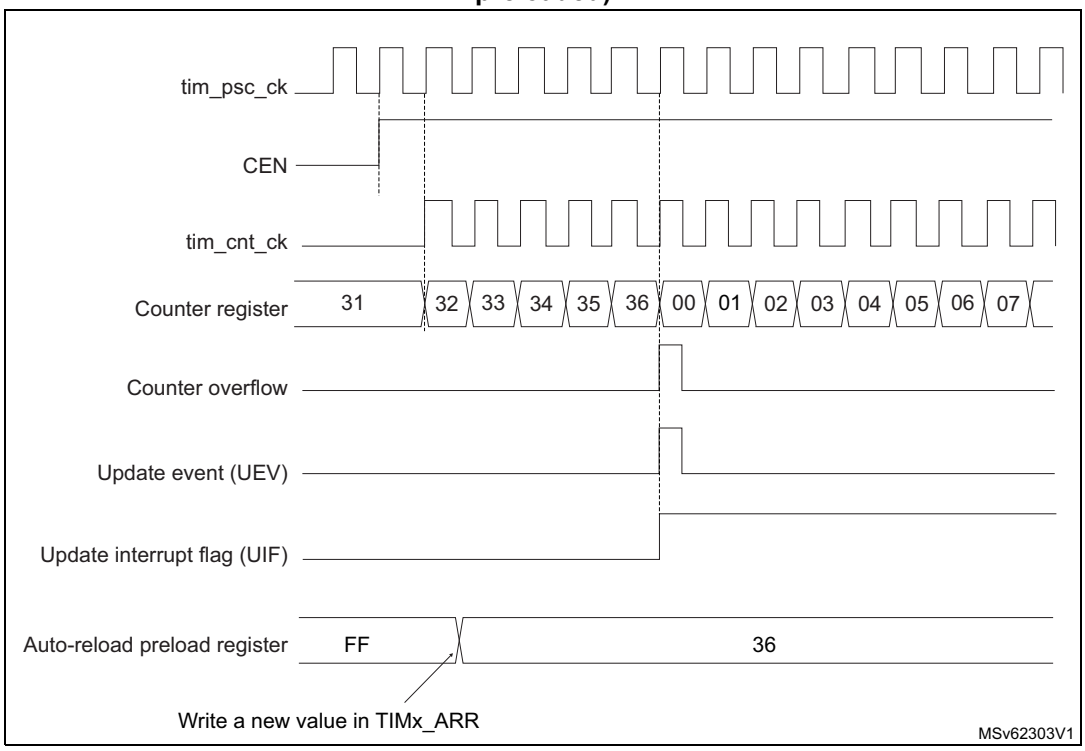
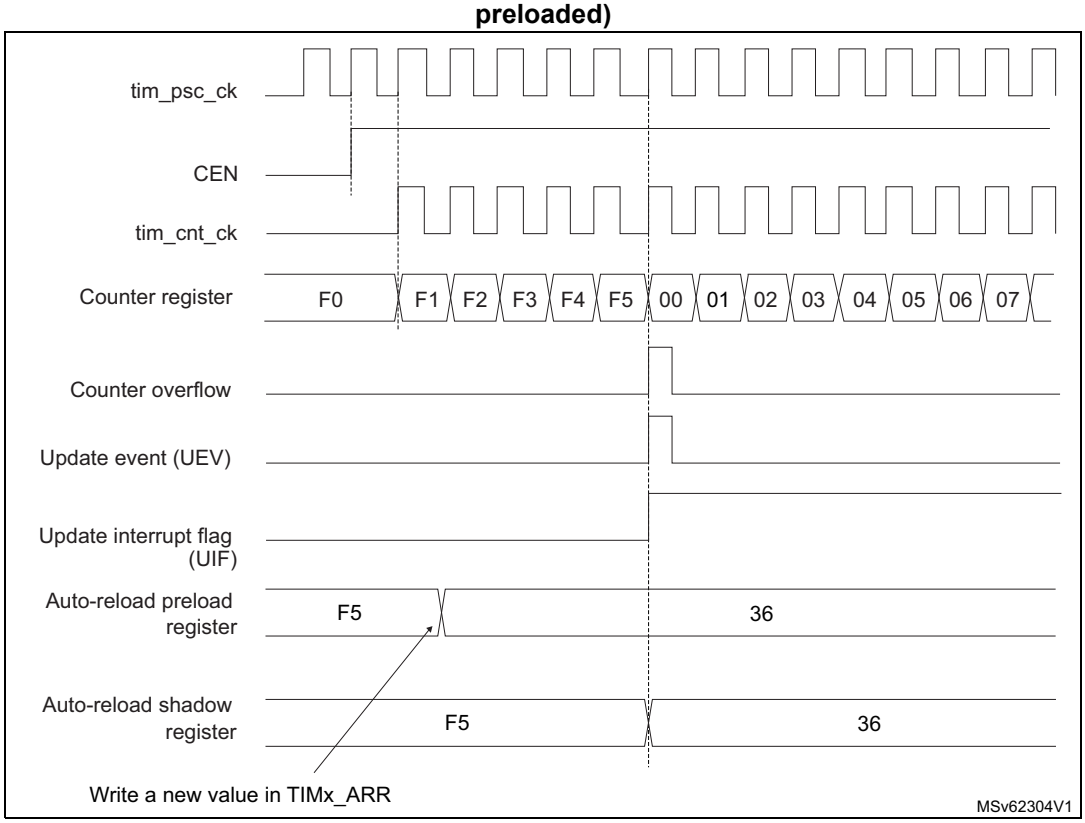


Figure 603. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



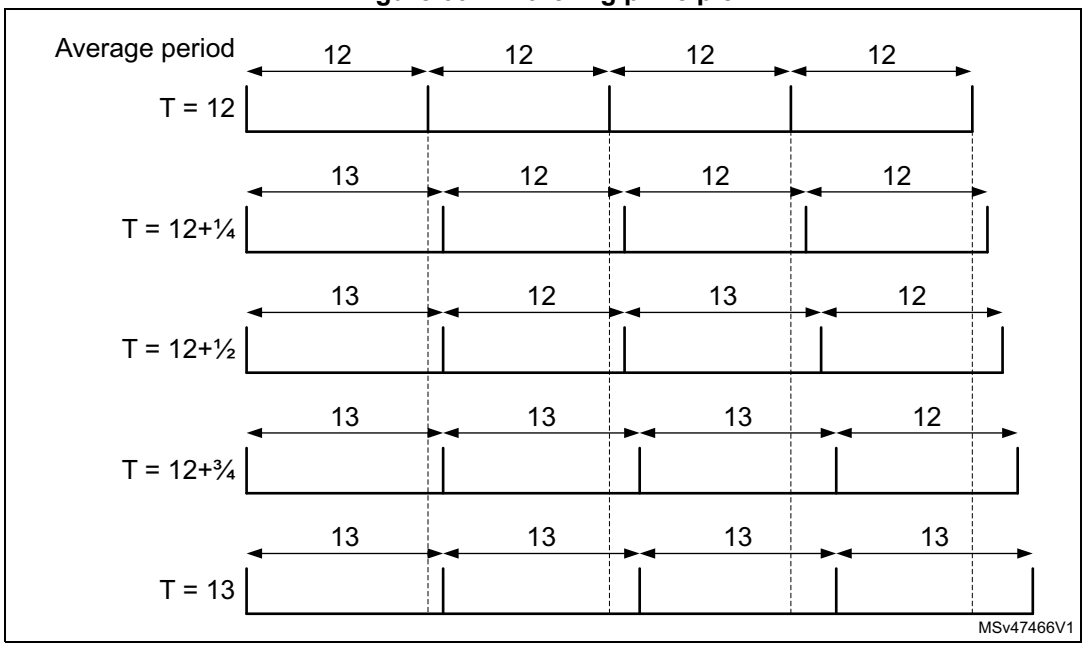
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

The figure below presents the dithering principle applied to 4 consecutive counting periods.

Figure 604. Dithering principle



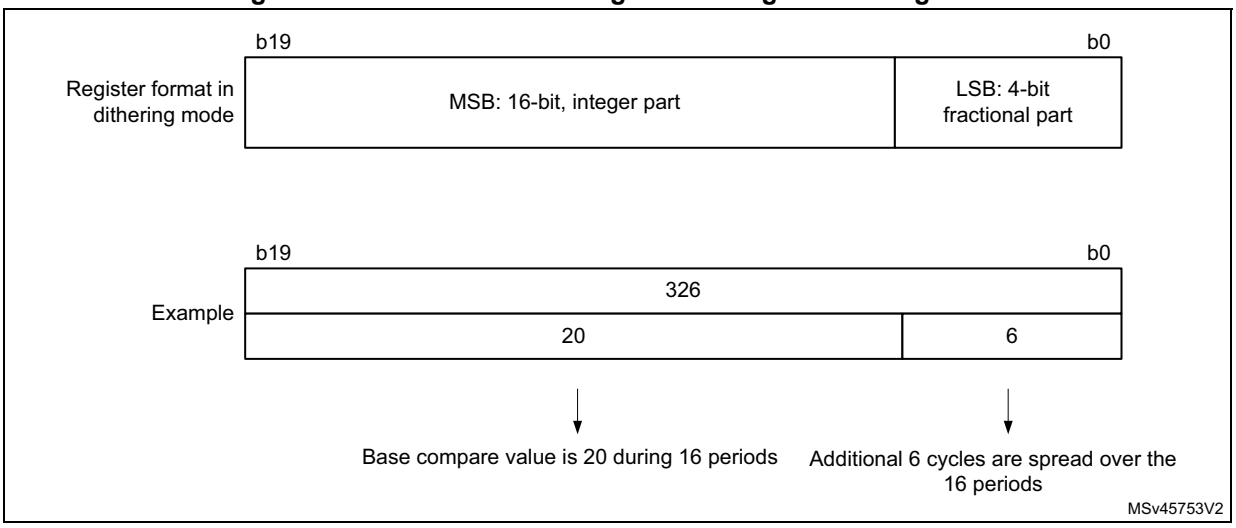
When the dithering mode is enabled, the register coding is changed as follows (refer to *Figure 605* for example):

- The 4 LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The ARR values are updated automatically if the DITHEN bit is set / reset (for instance, if ARR= 0x05 with DITHEN = 0, it is updated to ARR = 0x50 with DITHEN = 1).The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset*
2. *The ARR[3:0] bits must be reset*
3. *The DITHEN bit must be reset*
4. *The CEN bit can be set (eventually with ARPE = 1)*

Figure 605. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{Cnt}} \Rightarrow F_{CntMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

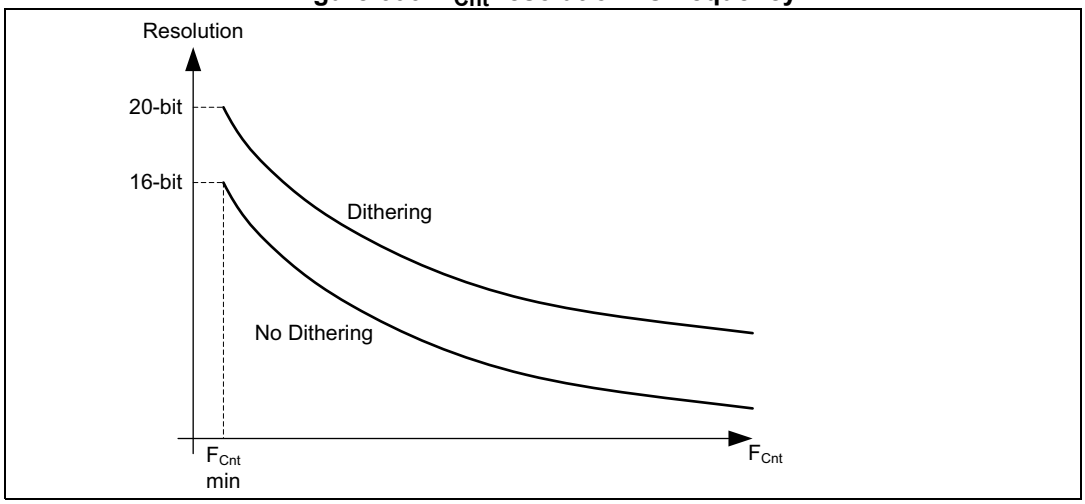
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR value is limited to 0xFFFEF in dithering mode (it corresponds to 65534 for the integer part and 15 for the dithered part).

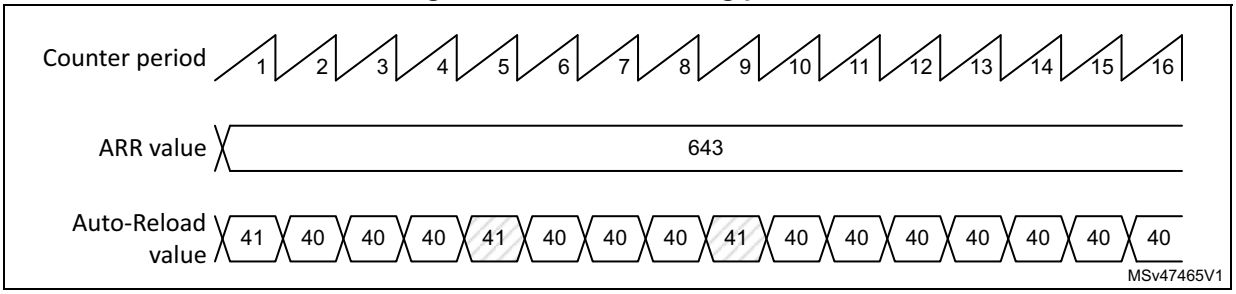
As shown in the figure below, the dithering mode allows to increase the PWM resolution whatever the PWM frequency is.

Figure 606. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 607* below.

Figure 607. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 437*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 437. TIMx_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

37.3.6 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions

caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

37.3.7 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 438*.

Table 438. DMA request

DMA acronym	DMA request	Enable control bit
TIM6_UP TIM7_UP	Update	UDE

37.3.8 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section debug support (DBG).

37.3.9 TIM6/TIM7 low-power modes

Table 439. Effect of low-power modes on TIM6/TIM7

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.

37.3.10 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 440*.

Table 440. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
TIM6 TIM7	Update	UIF	UIE	Write 0 in UIF	Yes

37.4 TIM6/TIM7 registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

37.4.1 TIM6/TIM7 memory map

Table 441. TIM6/TIM7 memory map

Offset	Register name
0x00	*TIMx control register 1 (TIMx_CR1)(x = 6 to 7)*
0x04	*TIMx control register 2 (TIMx_CR2)(x = 6 to 7)*
0x0C	*TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)*
0x10	*TIMx status register (TIMx_SR)(x = 6 to 7)*
0x14	*TIMx event generation register (TIMx_EGR)(x = 6 to 7)*
0x24	*TIMx counter (TIMx_CNT)(x = 6 to 7)*
0x28	*TIMx prescaler (TIMx_PSC)(x = 6 to 7)*
0x2C	*TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)*

37.4.2 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

37.4.3 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (tim_trgo).

001: **Enable** - the counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The counter enable signal is generated when the CEN control bit is written.

010: **Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or the peripheral receiving the tim_trgo must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

37.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.

1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.

1: Update interrupt enabled.

37.4.5 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– On counter overflow if UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

37.4.6 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

37.4.7 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0): The register holds the counter value.

Dithering mode (DITHEN = 1):

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

37.4.8 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim_cnt_ck}$ is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event, including when the counter is cleared through UG bit of TIMx_EGR register.

37.4.9 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 37.3.4: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0):

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

Dithering mode (DITHEN = 1):

The register holds the integer part in ARR[19:4]. The ARR[3:0] bit field contains the dithered part.

38 General-purpose timers (TIM15/TIM16)

38.1 TIM15/TIM16 introduction

The TIM15/TIM16 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 38.4.25: Timer synchronization (TIM15)*.

38.2 TIM15 main features

TIM15 includes the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

38.3 TIM16 main features

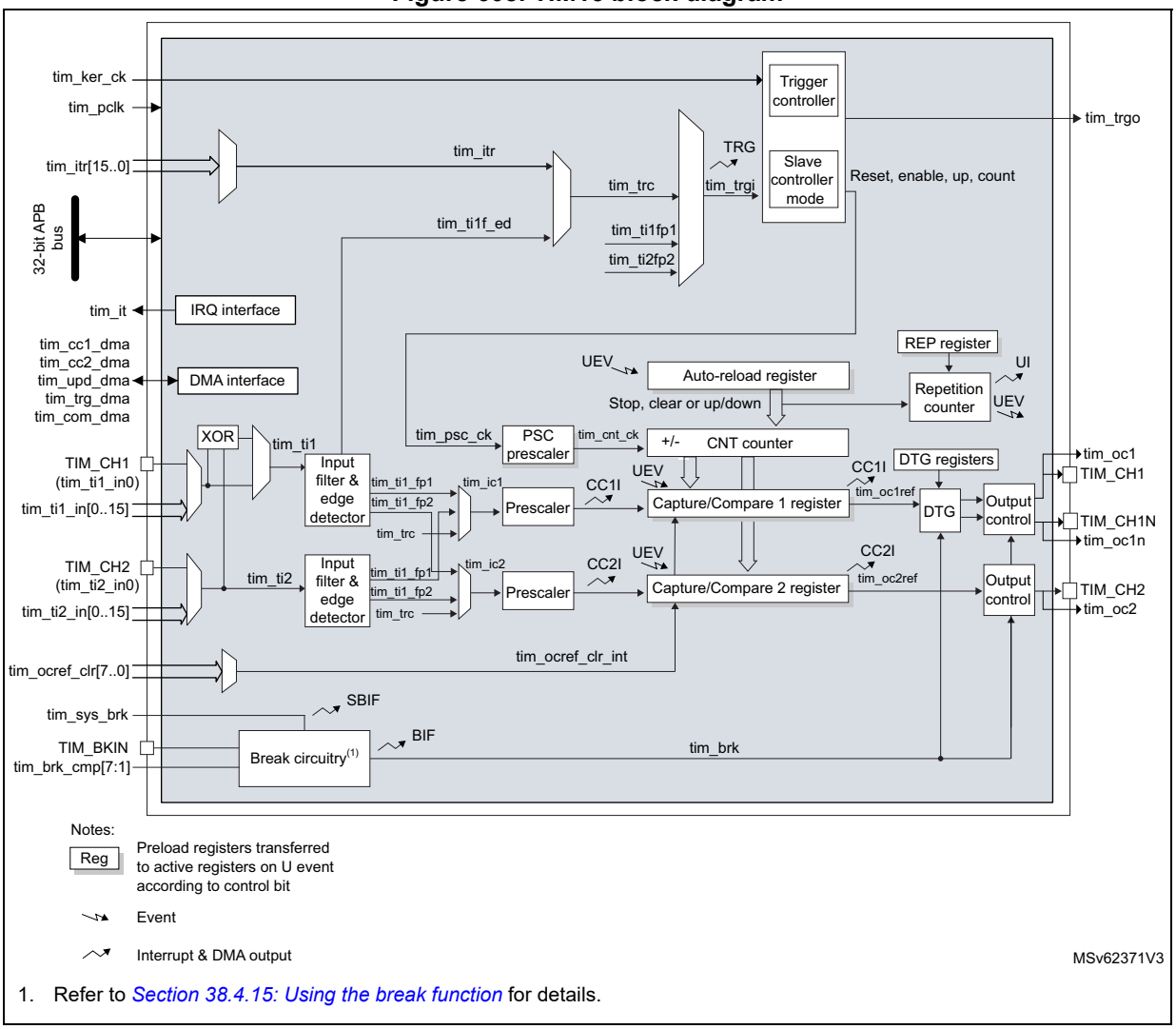
The TIM16 timers include the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

38.4 TIM15/TIM16 functional description

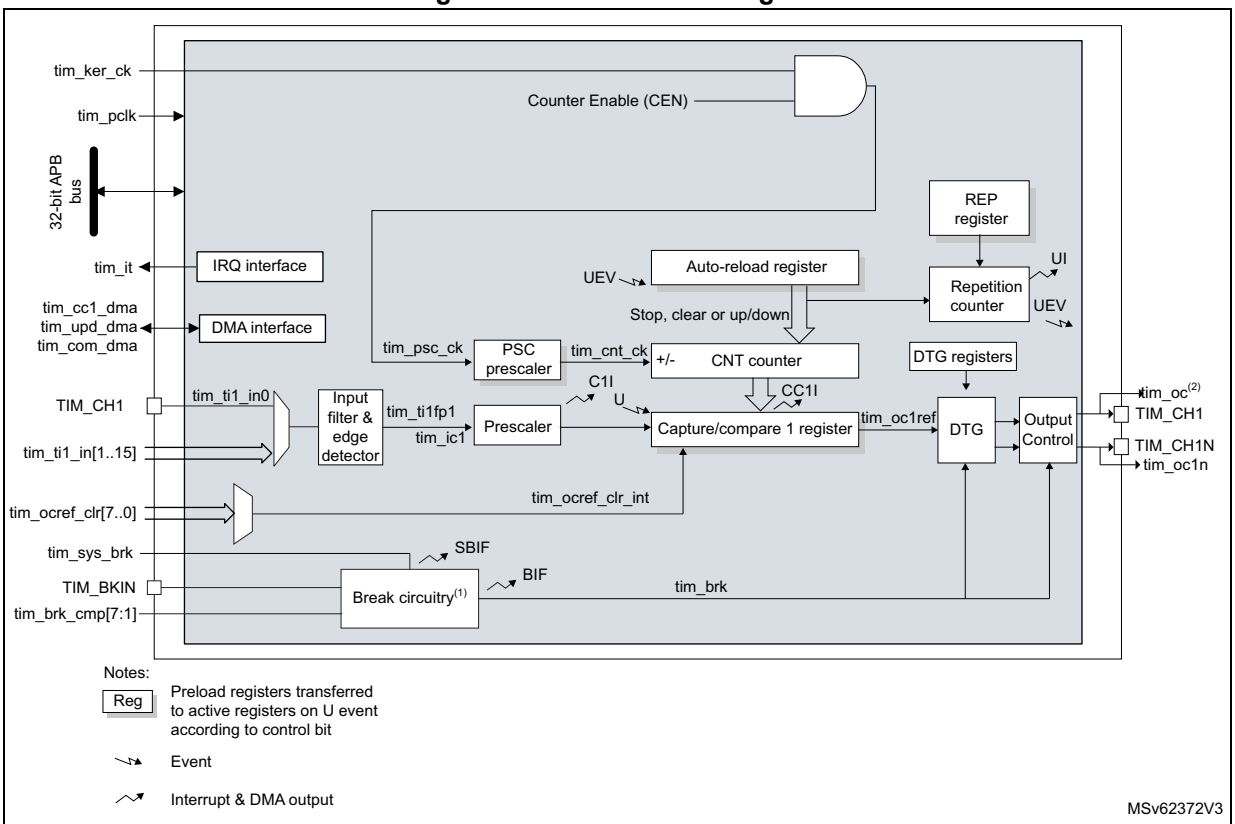
38.4.1 Block diagram

Figure 608. TIM15 block diagram



1. Refer to *Section 38.4.15: Using the break function* for details.

Figure 609. TIM16 block diagram



1. Refer to *Section 38.4.15: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (refer to internal trigger connection table in next section). Refer to *Section 38.4.26: Using timer output as trigger for other timers (TIM16)* for details.

38.4.2 TIM15/TIM16 pins and internal signals

Table 442 and *Table 443* in this section summarize the TIM inputs and outputs.

Table 442. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel can be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have dead time insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

1. Available for TIM15 only.

Table 443. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[7:1]	Input	Break input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma tim_cc2_dma(1)	Output	Timer capture / compare 1...2 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trg_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

1. Available for TIM15 only.

The sources connected to the tim_ti[1..2] input multiplexers are listed in the *Section 5.5.5.1: Interconnects to the tim_ti[1..4] input multiplexers* from the Device configuration chapter.

The internal sources connected to the tim_itr input multiplexer are listed in the *Section 5.5.5.2: Interconnects to the trigger input multiplexers* from the Device configuration chapter.

The sources connected to the tim_brk input are listed in the *Section 5.5.5.3: Interconnects to the break inputs* from the Device configuration chapter.

The internal sources connected to the tim_ocref_clr input multiplexer are listed in the *Section 5.5.5.4: Interconnects to the ocref_clr input multiplexer* from the Device configuration chapter.

38.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The contents of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 610 and *Figure 611* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 610. Counter timing diagram with prescaler division change from 1 to 2

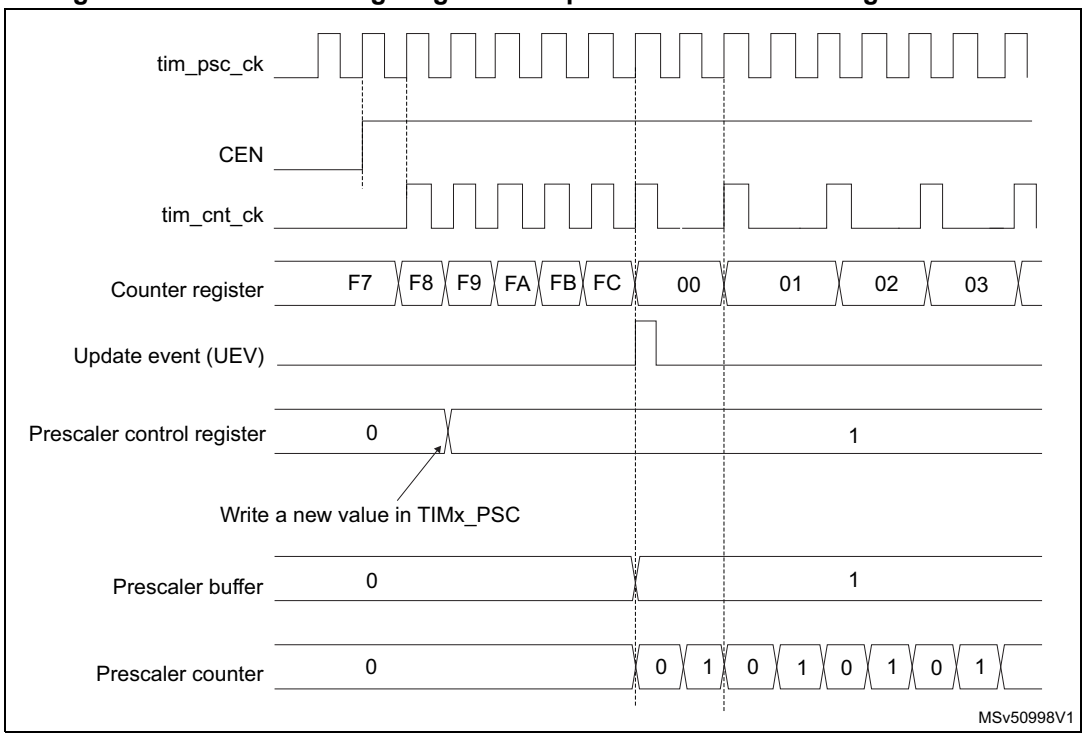
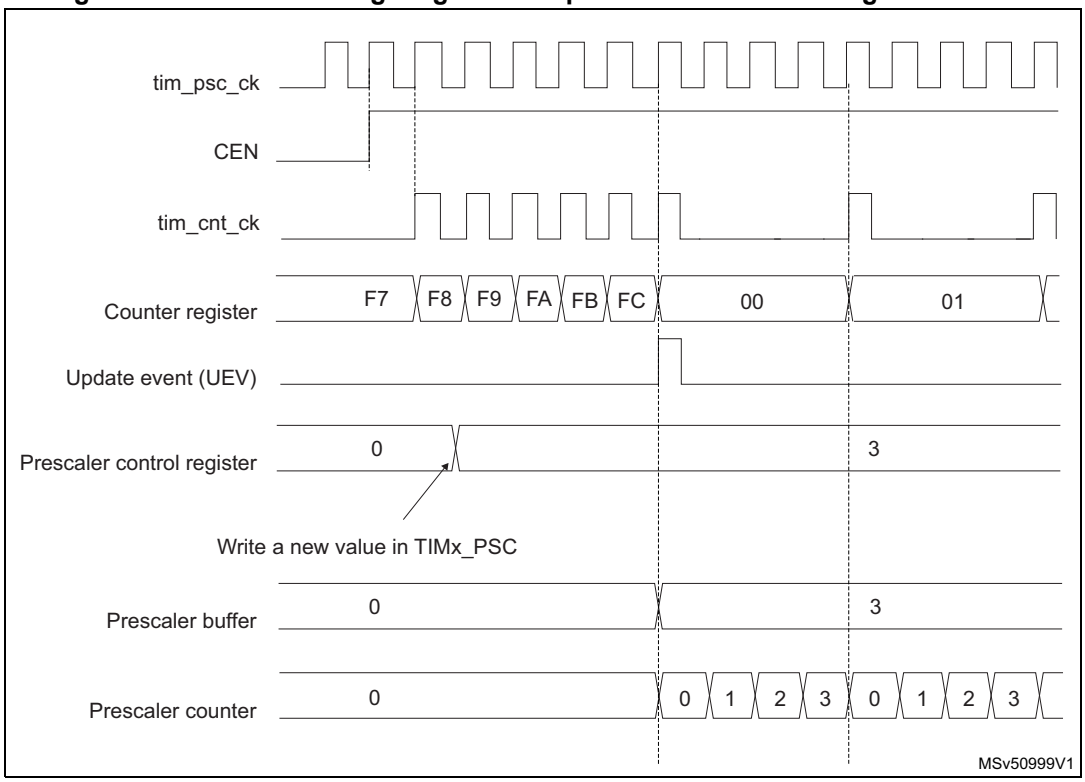


Figure 611. Counter timing diagram with prescaler division change from 1 to 4



38.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 612. Counter timing diagram, internal clock divided by 1

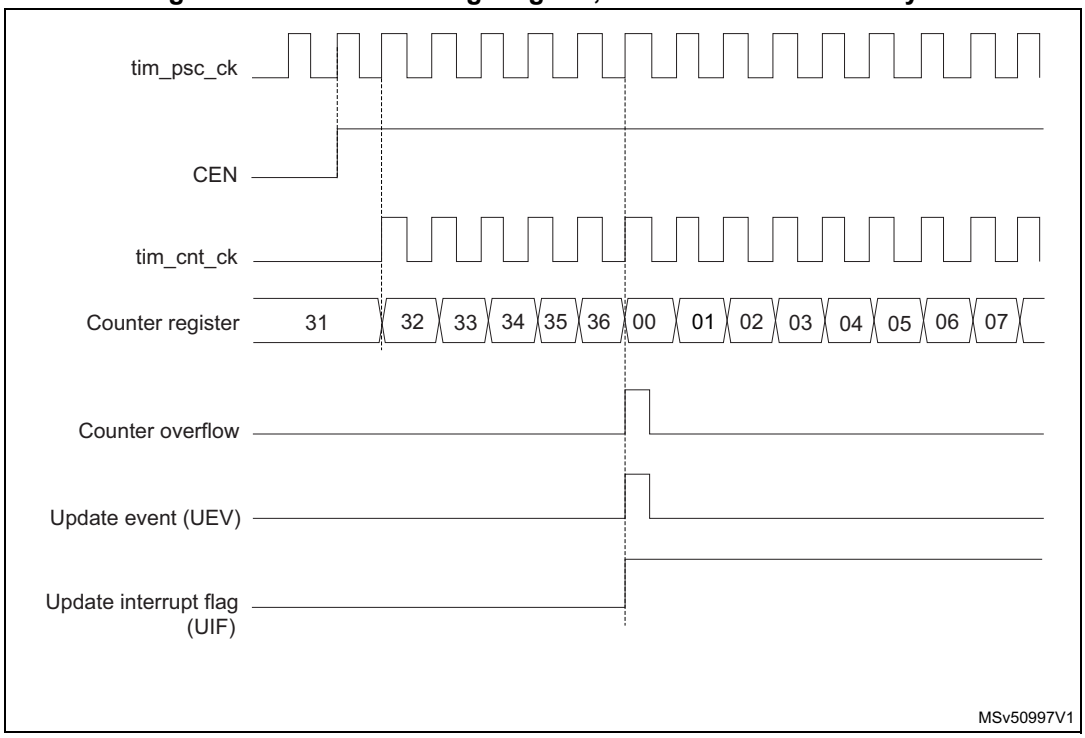


Figure 613. Counter timing diagram, internal clock divided by 2

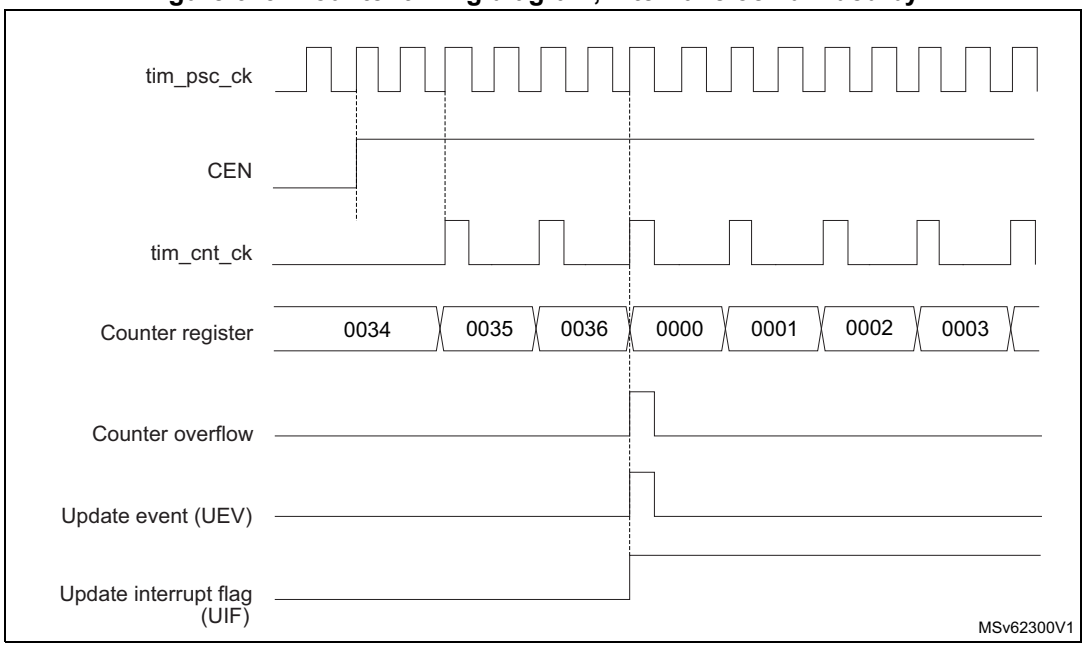


Figure 614. Counter timing diagram, internal clock divided by 4

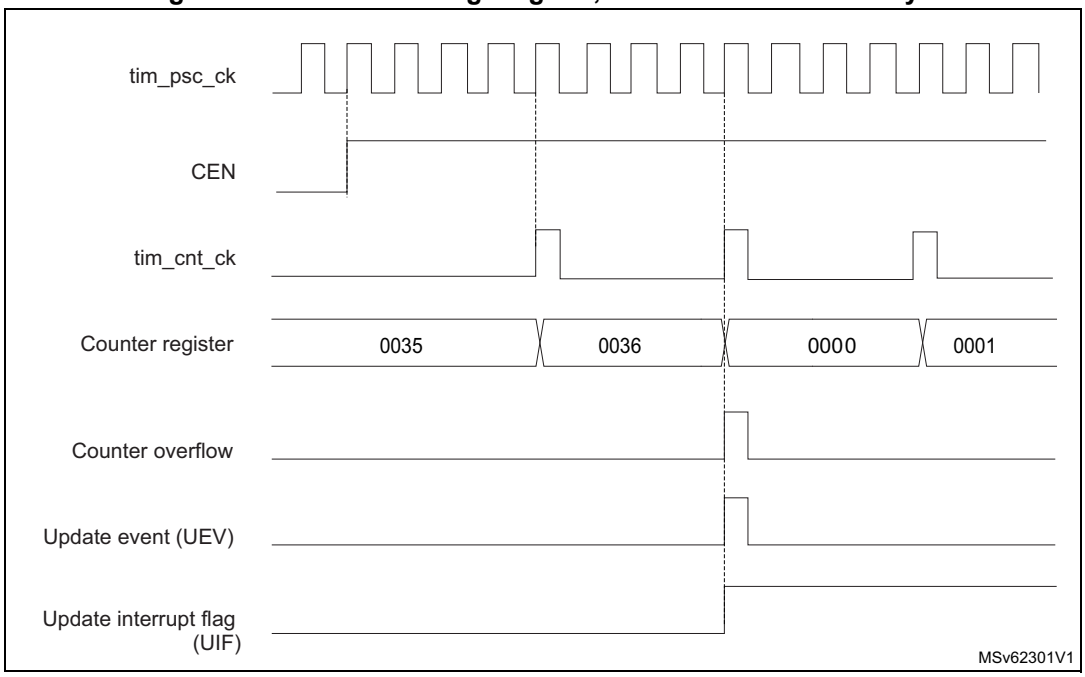


Figure 615. Counter timing diagram, internal clock divided by N

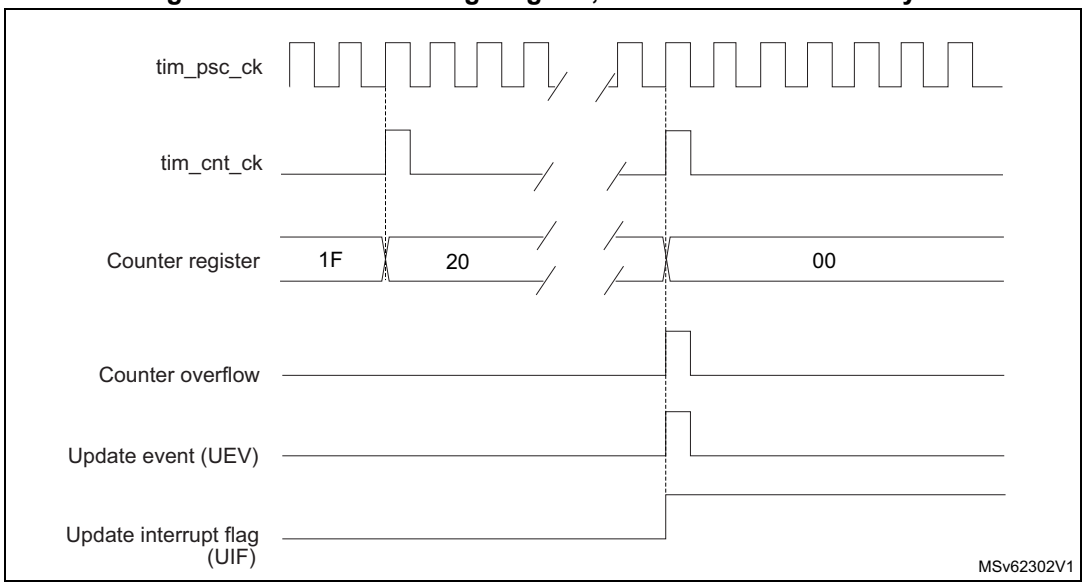


Figure 616. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

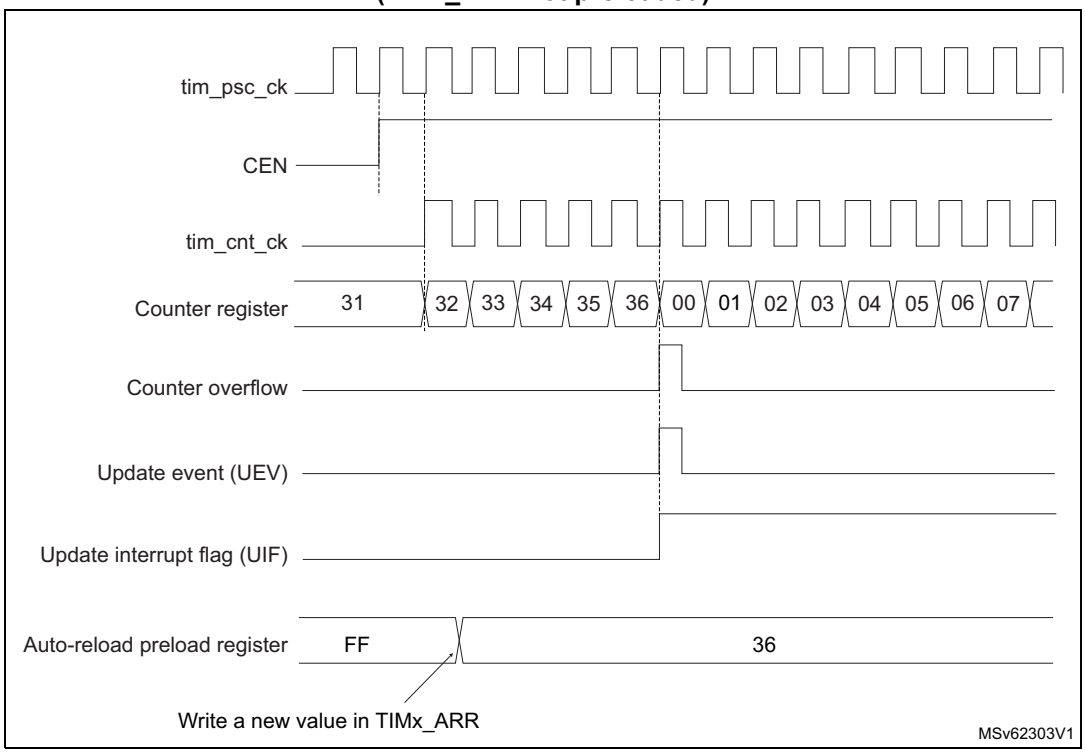
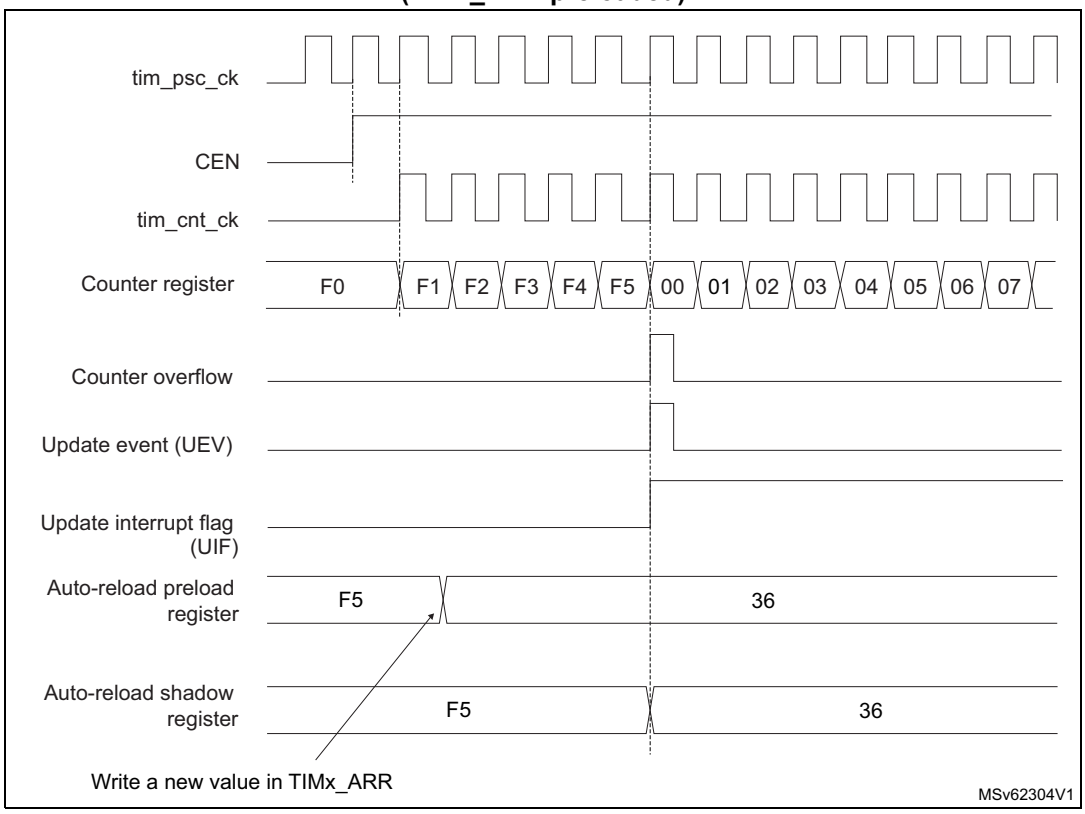


Figure 617. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



38.4.5 Repetition counter

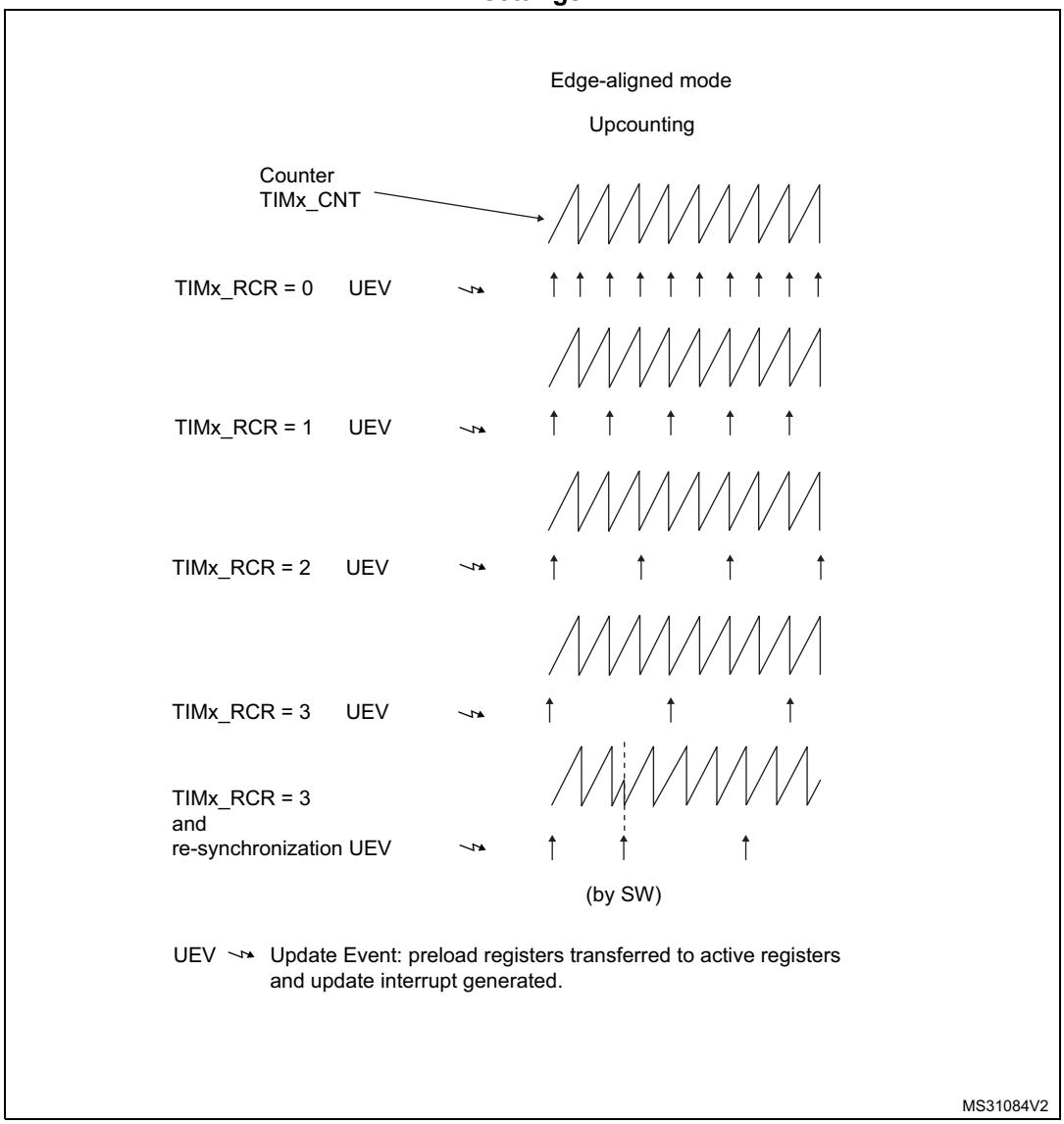
Section 38.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 618*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 618. Update rate examples depending on mode and TIMx_RCR register settings



38.4.6 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to section *Using one timer to enable another timer* for more details.

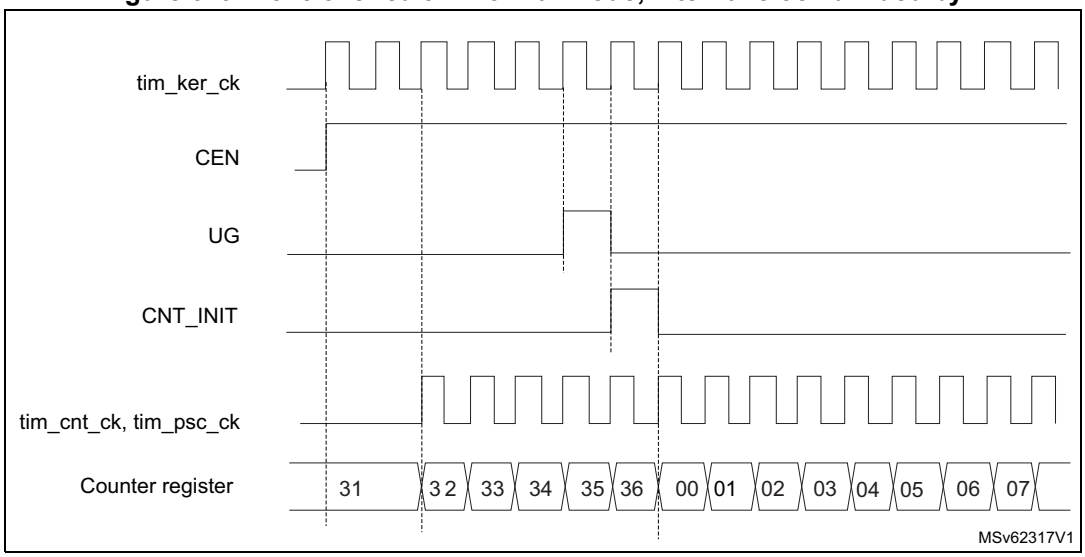
Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 619 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

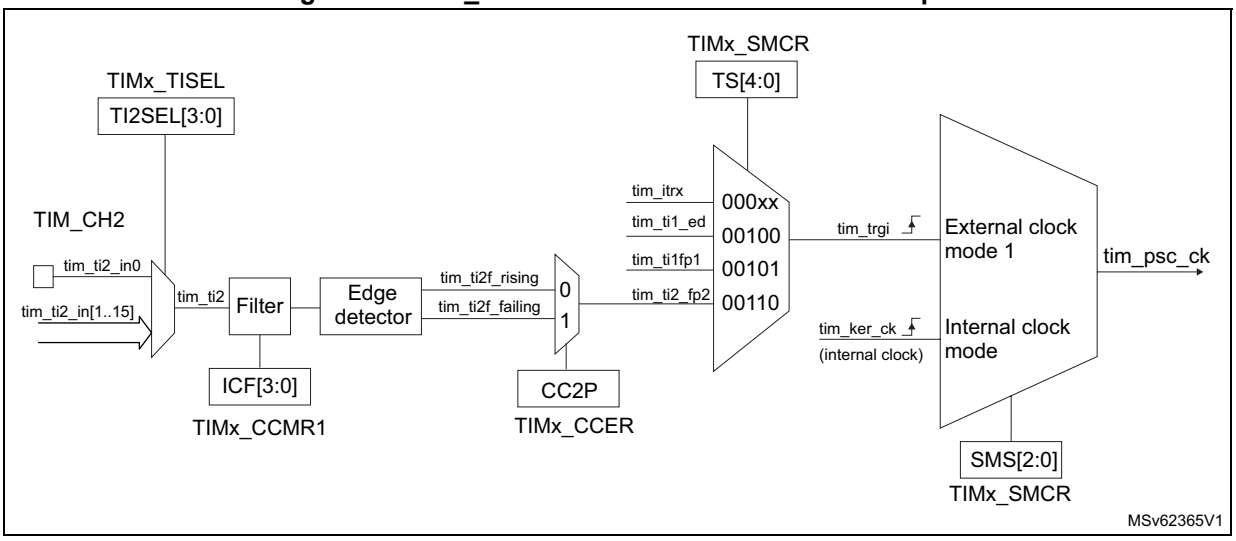
Figure 619. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 620. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

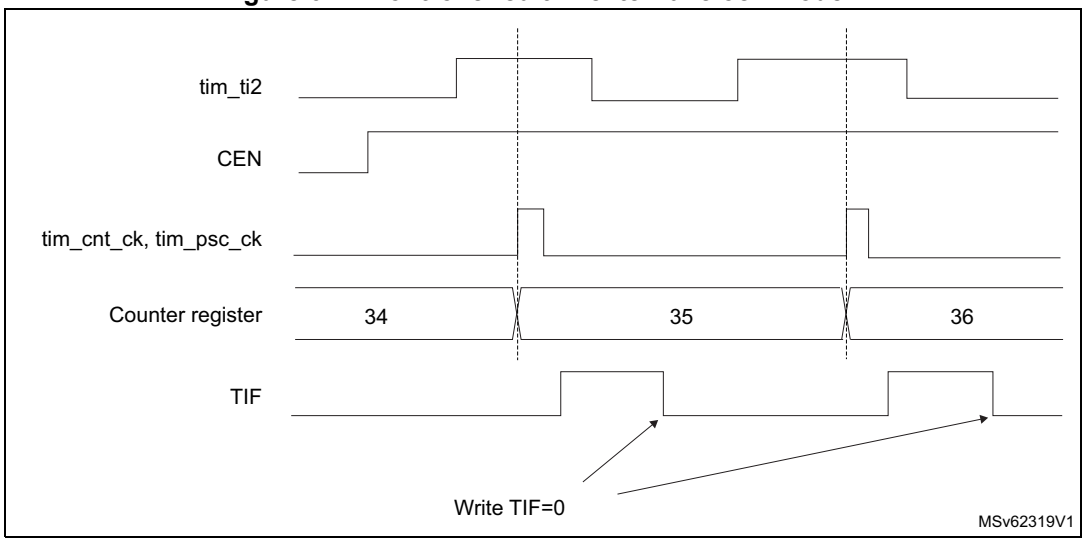
1. Select the proper tim_ti2_in[0..15] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 621. Control circuit in external clock mode 1



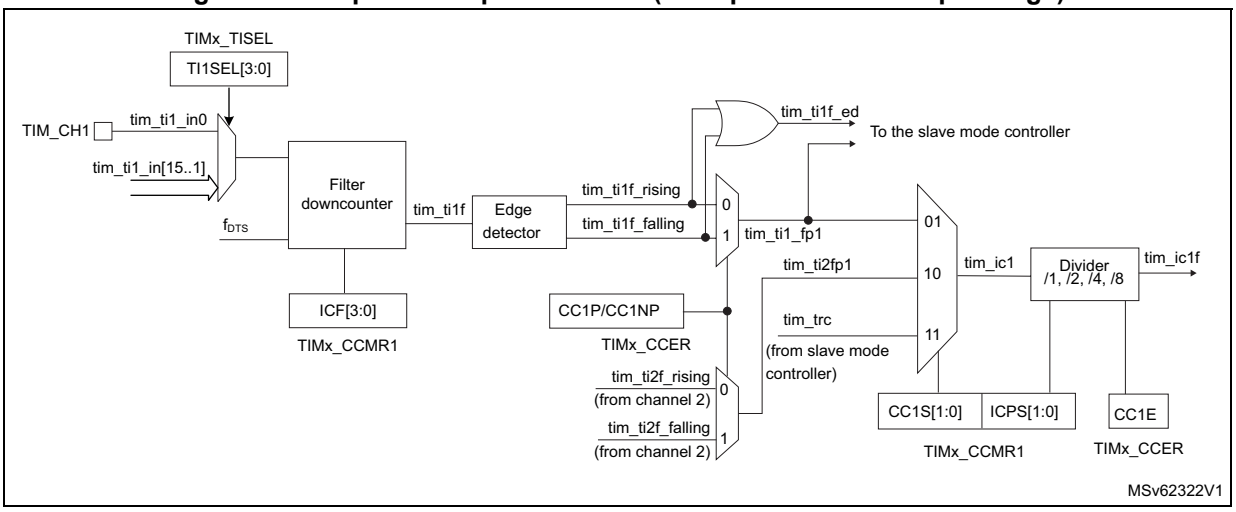
38.4.7 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 622 to *Figure 625* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 622. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 623. Capture/compare channel 1 main circuit

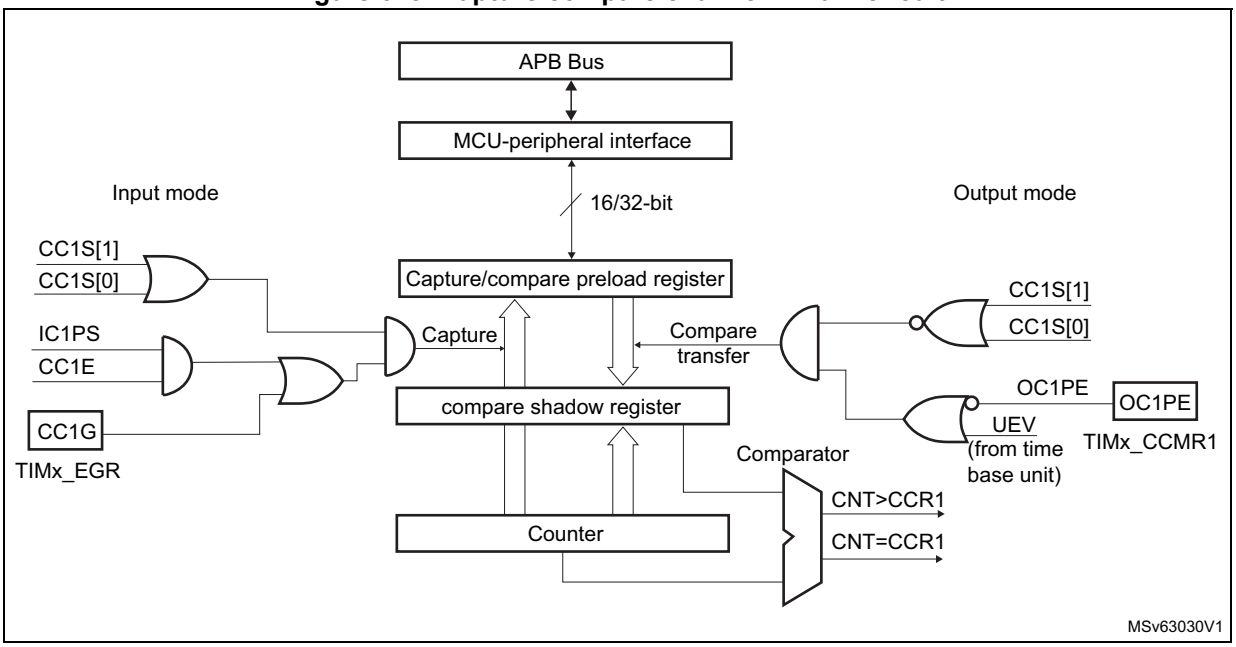


Figure 624. Output stage of capture/compare channel (channel 1)

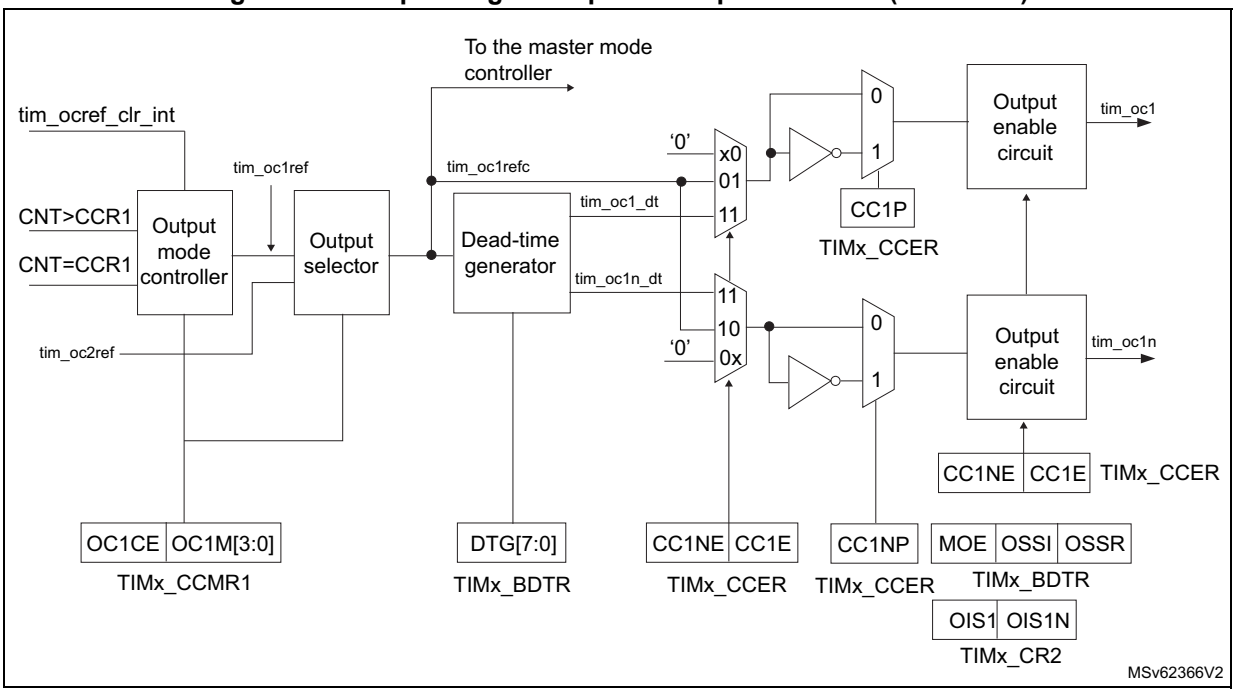
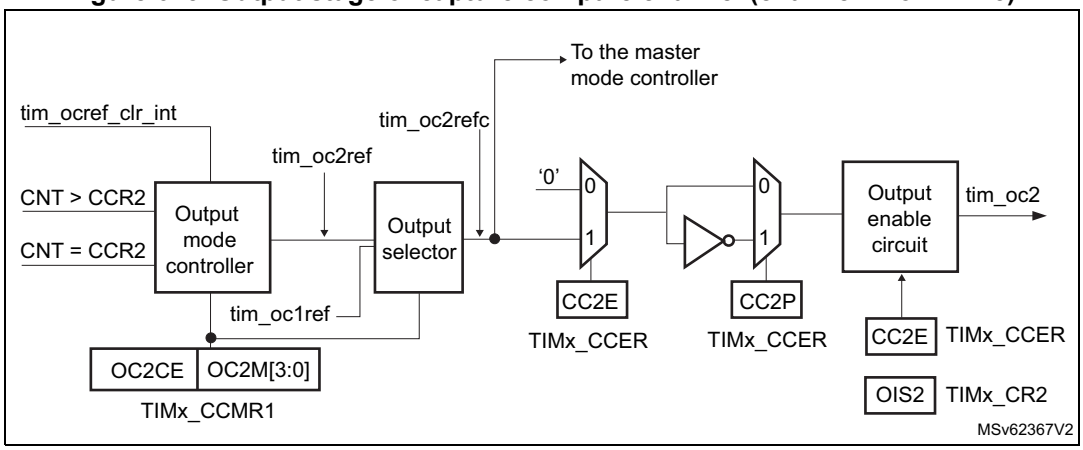


Figure 625. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

38.4.8 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was

already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[1..15] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_ti1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

38.4.9 PWM input mode (only for TIM15)

This mode allows to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIMx_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

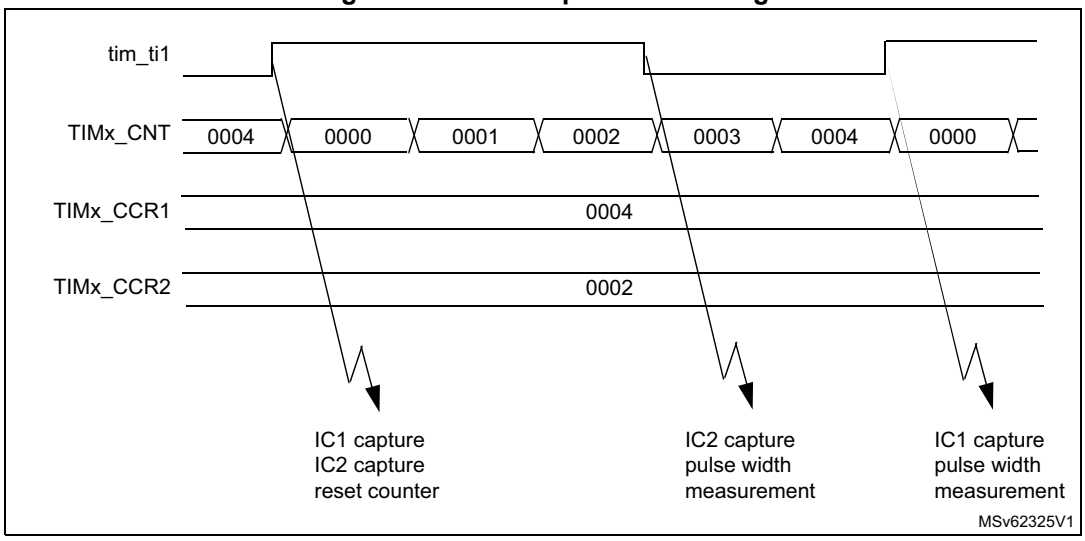
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These 2 tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[0.15] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to ‘10’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 626. PWM input mode timing



Note: The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals.

38.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in *Section 38.4.11: Output compare mode*.

38.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

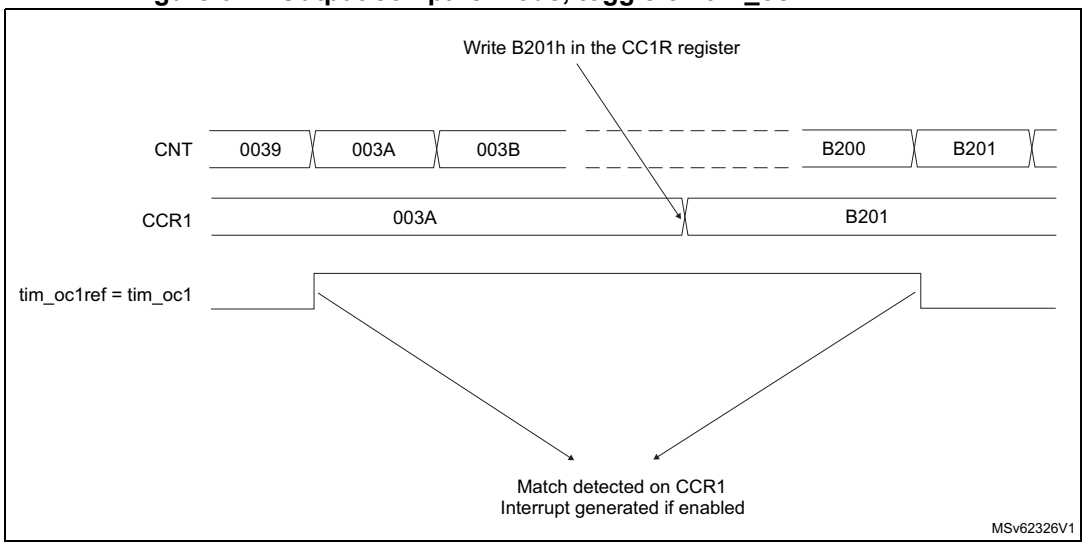
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 627*.

Figure 627. Output compare mode, toggle on tim_oc1



38.4.12 PWM mode

Pulse width modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

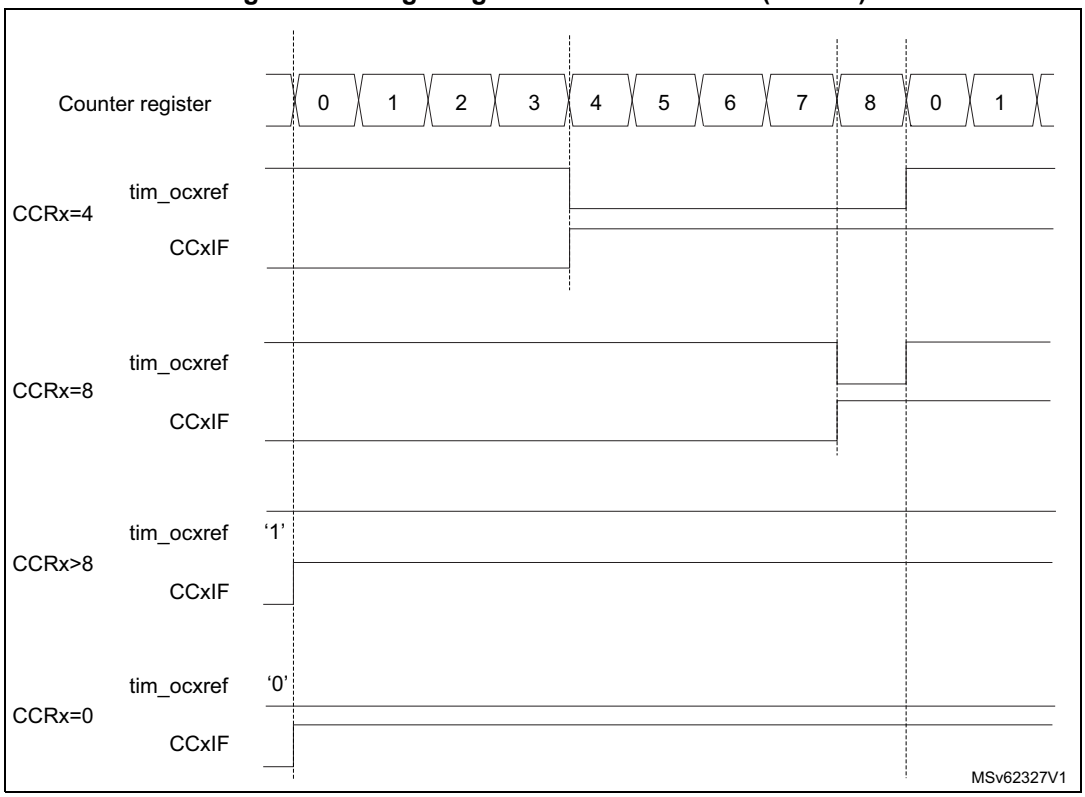
tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

TIM15/TIM16 are capable of upcounting only. Refer to the section *Upcounting mode*.

In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then tim_ocxref is held at ‘1’. If the compare value is 0 then tim_ocxref is held at ‘0’. *Figure 628* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 628. Edge-aligned PWM waveforms (ARR=8)

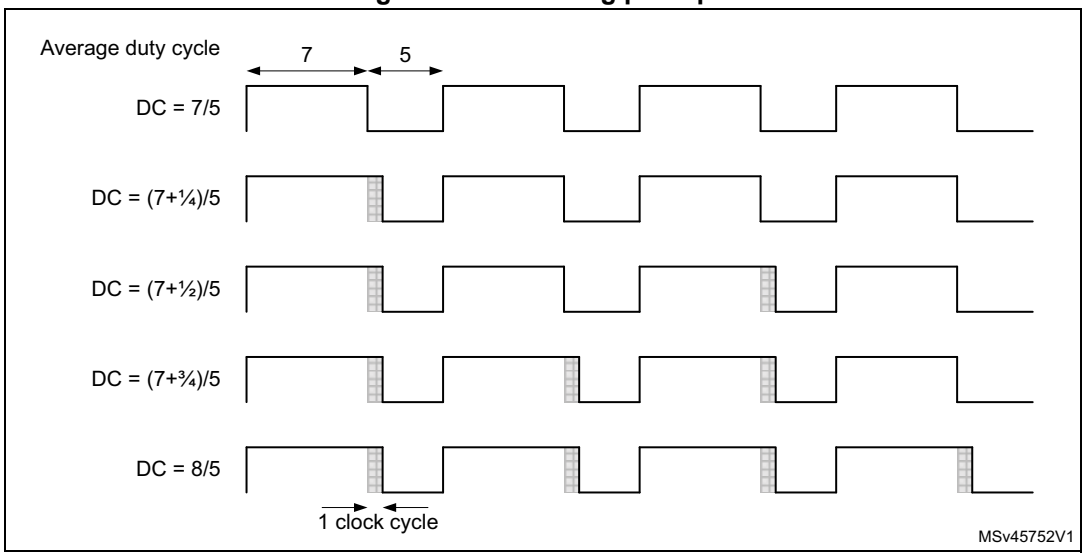


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 629* presents the dithering principle applied to 4 consecutive PWM cycles.

Figure 629. Dithering principle



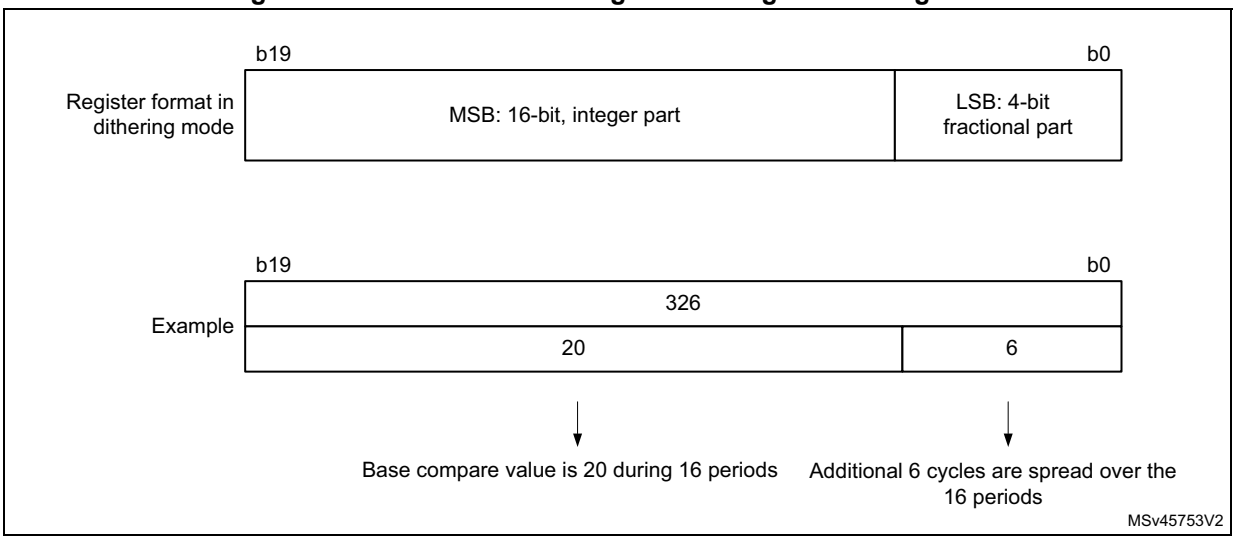
When the dithering mode is enabled, the register coding is changed as follows (refer to *Figure 630* for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- the MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The ARR and CCR values is updated automatically if the DITHEN bit is set / reset (for instance, if ARR= 0x05 with DITHEN=0, it is updated to ARR = 0x50 with DITHEN = 1). The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset*
2. *The ARR[3:0] bits must be reset*
3. *The DITHEN bit must be reset*
4. *The CCIF flags must be cleared*
5. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 630. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

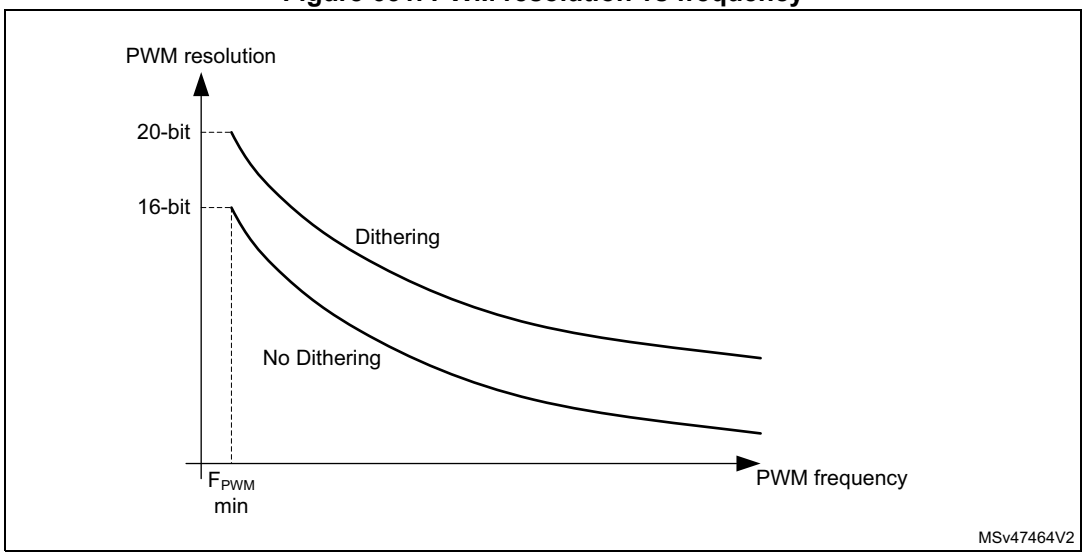
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

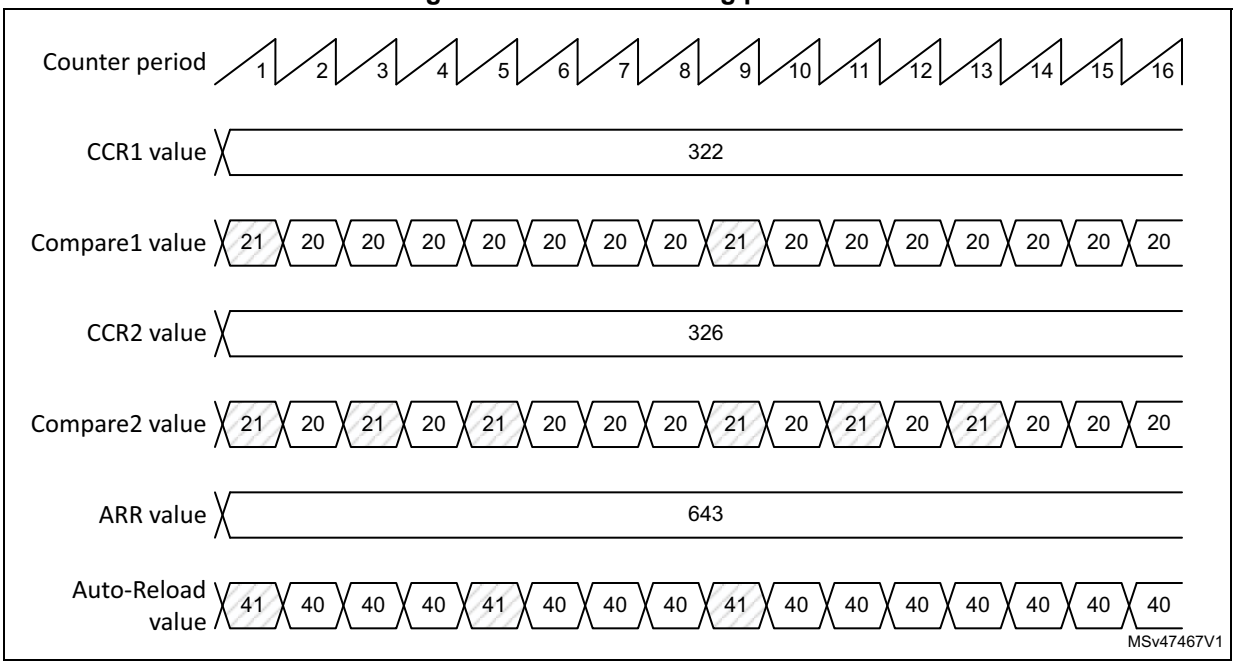
As shown in *Figure 631*, the dithering mode allows to increase the PWM resolution whatever the PWM frequency.

Figure 631. PWM resolution vs frequency



The duty cycle and / or period changes are spread over 16 consecutive periods, as described in *Figure 632*.

Figure 632. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in *Table 444*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 444. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 444. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

38.4.13 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently of two channels (one tim_ocx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

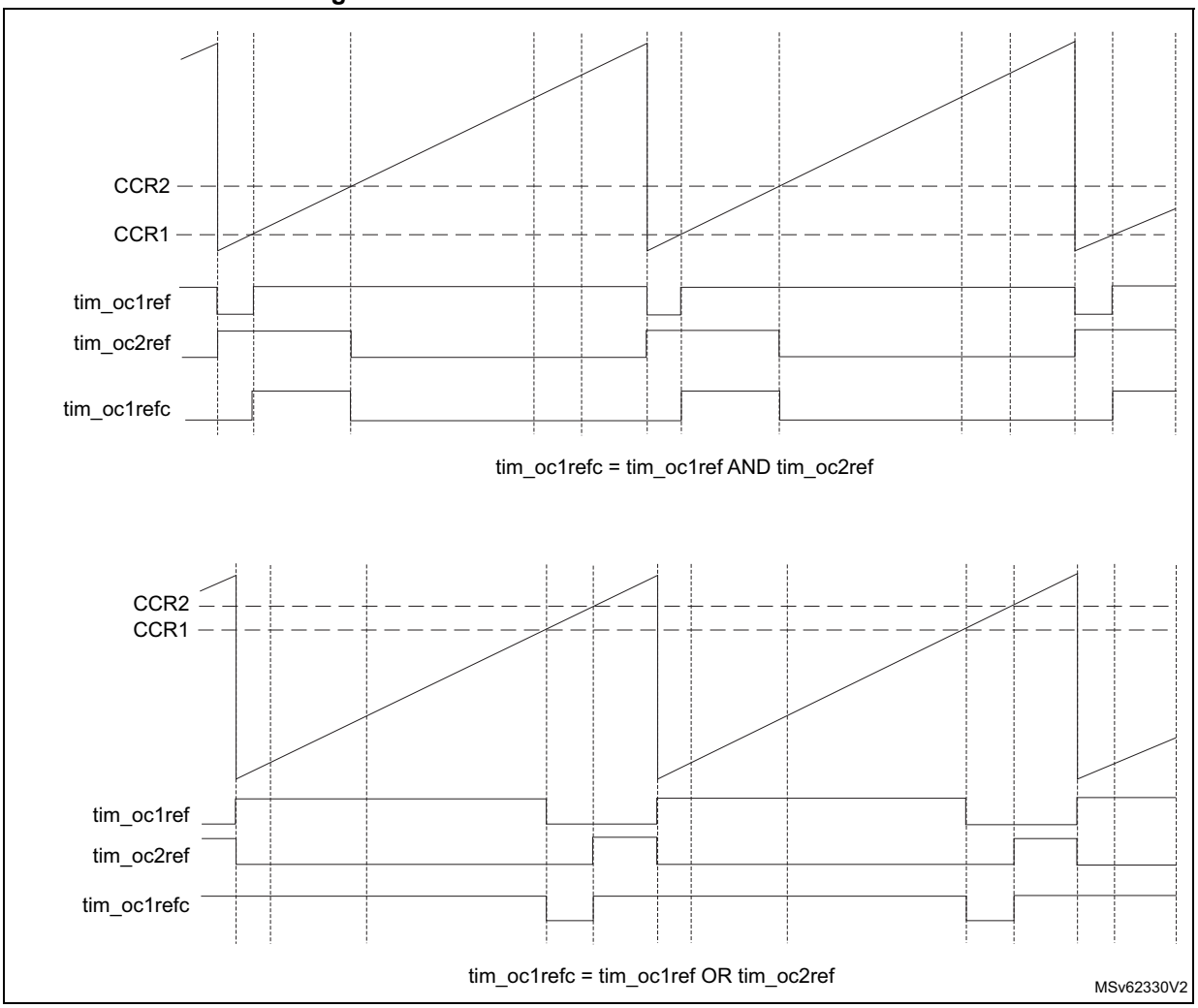
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 633 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- tim_oc1refc = tim_oc1ref AND tim_oc2ref signal, where:
  channel 1 is configured in Combined PWM mode 2
  channel 2 is configured in PWM mode 1
- tim_oc1refc = tim_oc1ref OR tim_oc2ref signal, where:
  channel 1 is configured in Combined PWM mode 1
  channel 2 is configured in PWM mode 2

Figure 633. Combined PWM mode on channel 1



38.4.14 Complementary outputs and dead-time insertion

The TIM15/TIM16 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 451* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

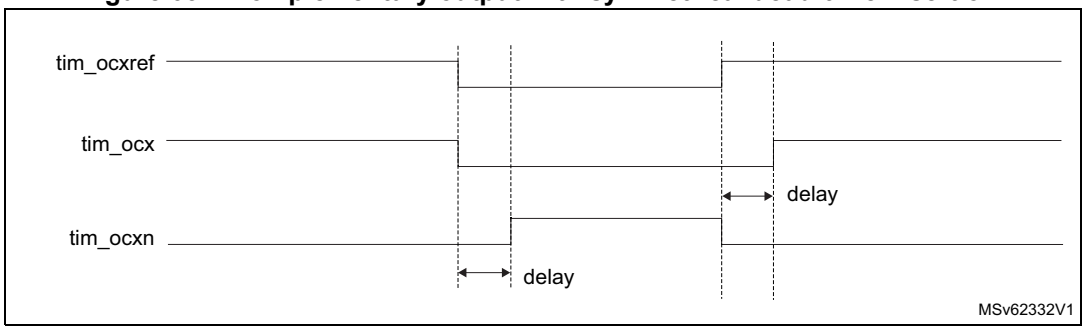
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates 2 outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 634. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIMx_DTR2 allows to differentiate the deadtime values for rising and falling edges of the reference signal, as shown in *Figure 635*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bit field in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bit field in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must be not modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bit field DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since the last update is discarded and previous value is used.

Figure 635. Asymmetrical deadtime

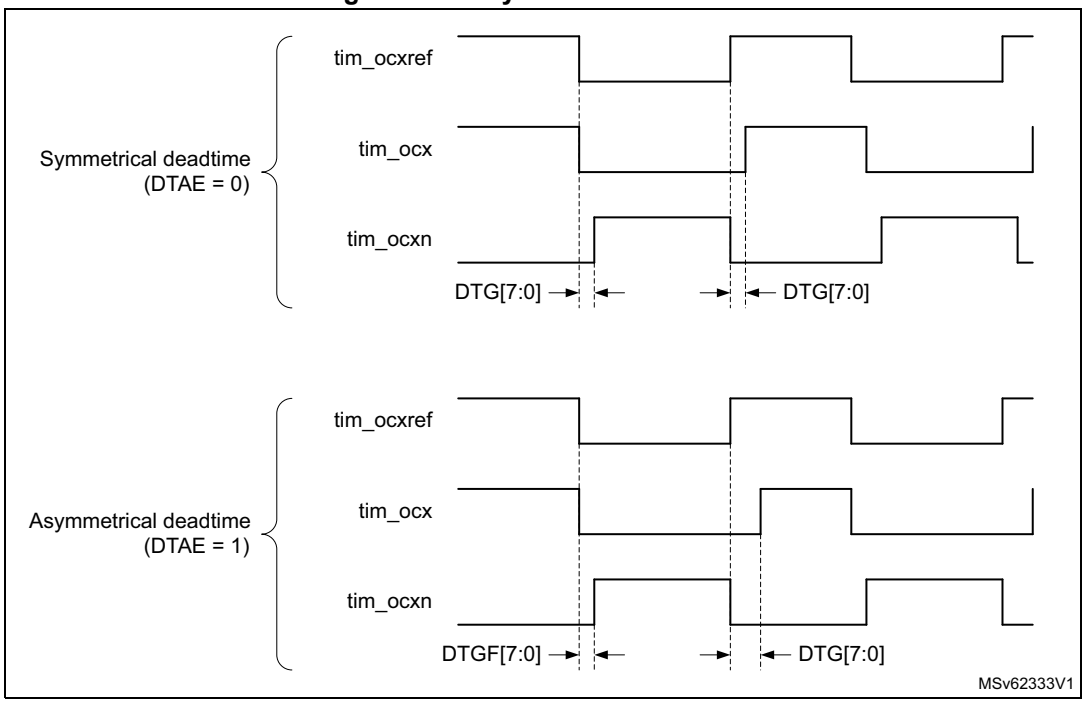


Figure 636. Dead-time waveforms with delay greater than the negative pulse.

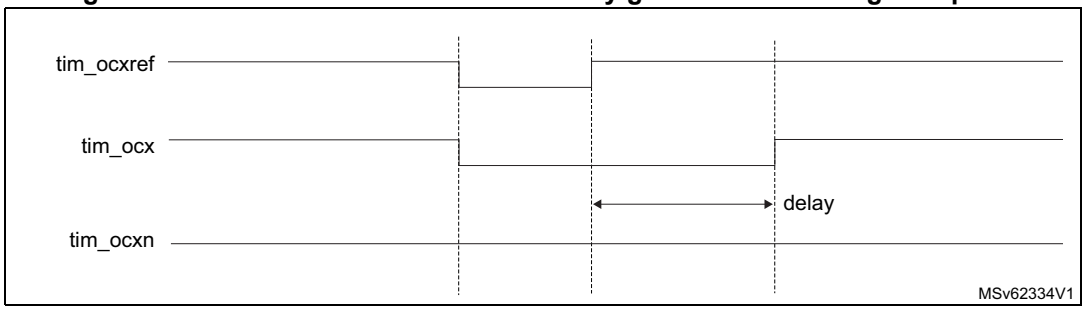
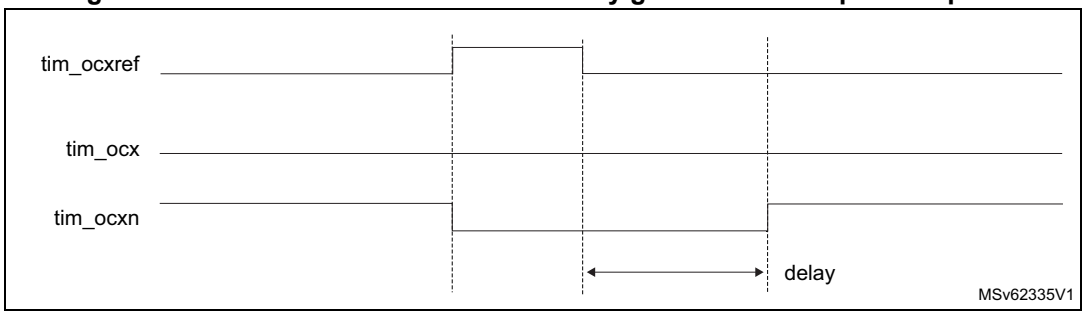


Figure 637. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to the *Section 38.9.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16)* for delay calculation.

Re-directing tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be re-directed to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP=0 then tim_ocxn=tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE=CCxNE=1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

38.4.15 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to the *Table 450: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which have:

- Programmable polarity (BKP bit in the TIMx_BDTR register).
- Programmable enable bit (BKE bit in the TIMx_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

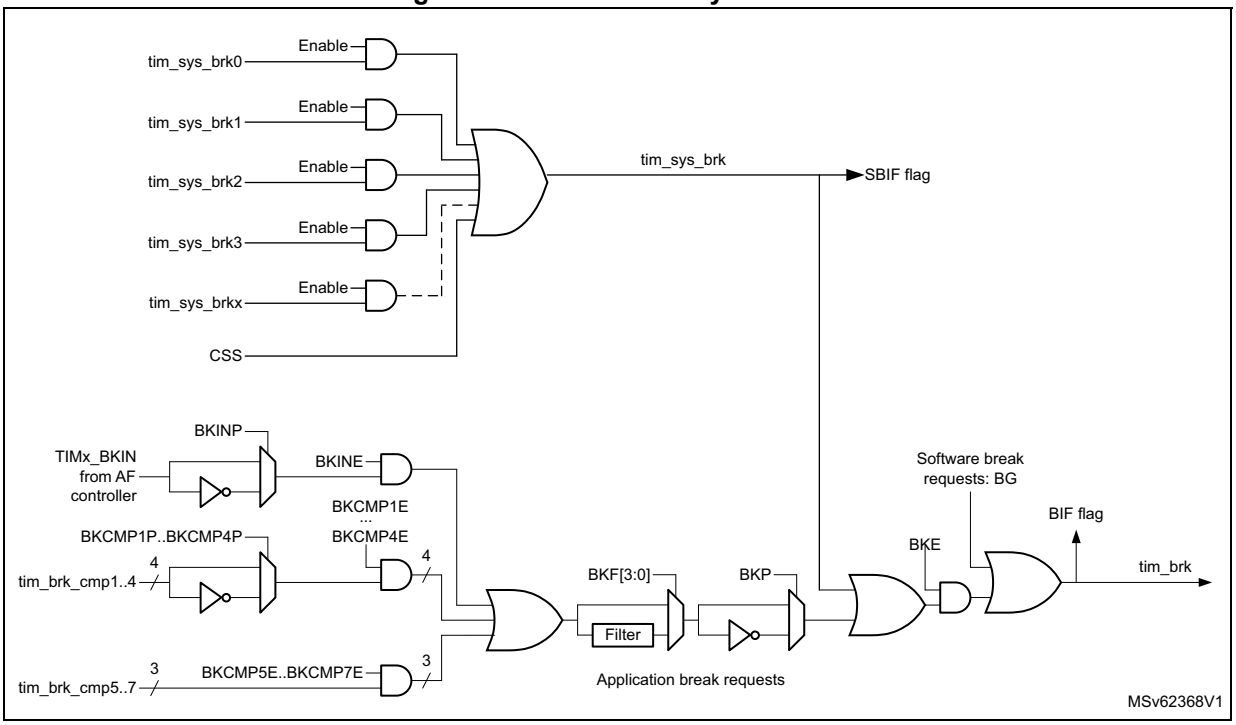
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering.
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to the *Section 38.4.2: TIM15/TIM16 pins and internal signals* for product specific implementation).
  - coming from a system break request on the tim_sys_brk inputs (refer to the *Section 38.4.2* for product specific implementation).

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 638*.

Figure 638. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the AFIO controller (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the AFIO controller) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).
  - If OSSI=0 then the timer releases the enable outputs (taken over by the AFIO controller which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

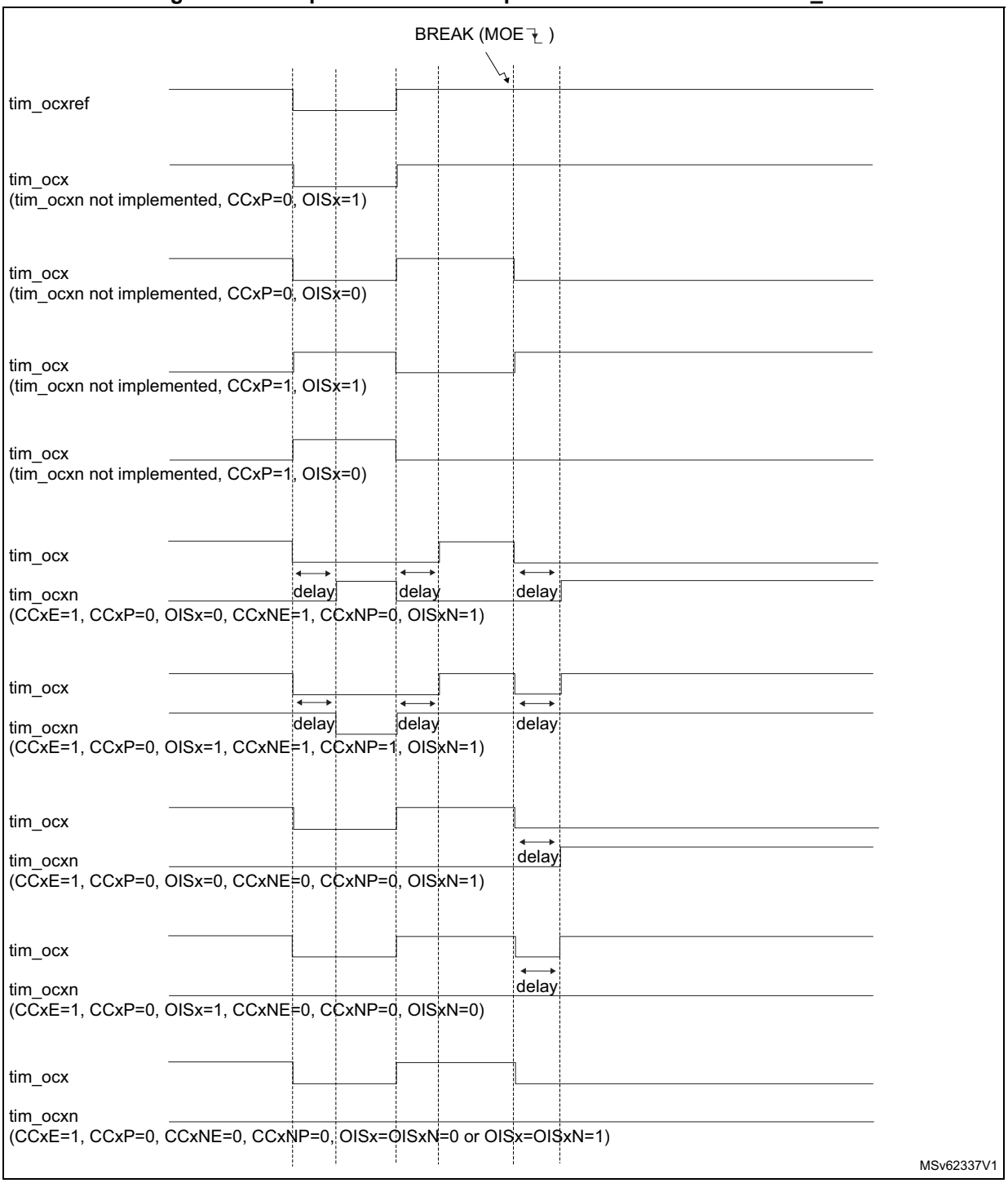
Note: The break input is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to the *Section 38.9.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16)*. The LOCK bits can be written only once after an MCU reset.

The figure below shows an example of behavior of the outputs in response to a break.

Figure 639. Output behavior in response to a break event on tim_brk



38.4.16 Bidirectional break input

The TIM15/TIM16 are featuring bidirectional break I/Os, as represented in *Figure 640*.

They allow to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example, CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output from being re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (refer to *Table 445*).

Table 445. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

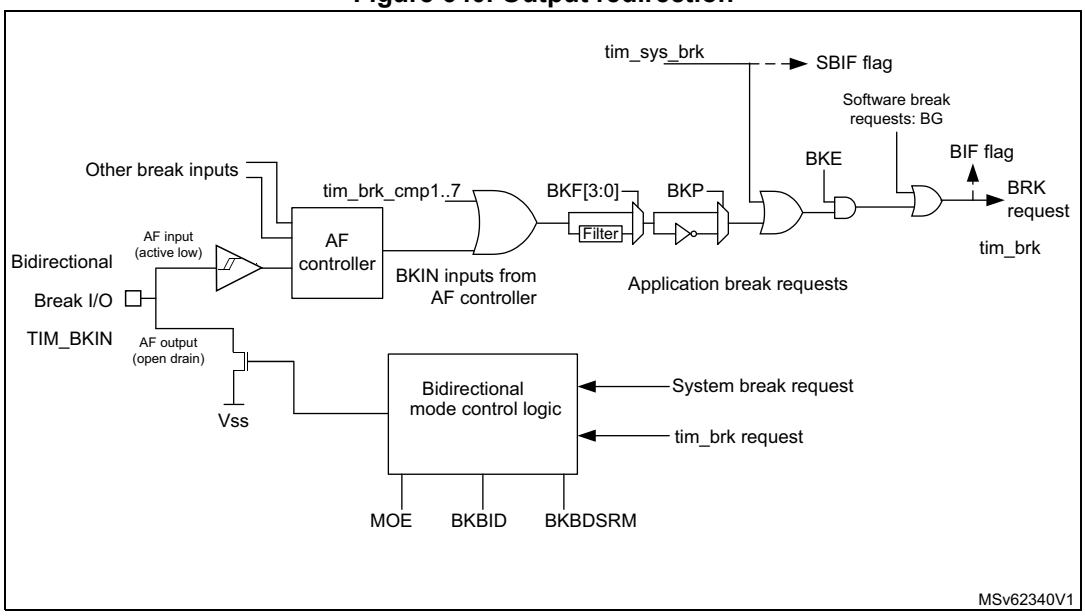
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 640. Output redirection

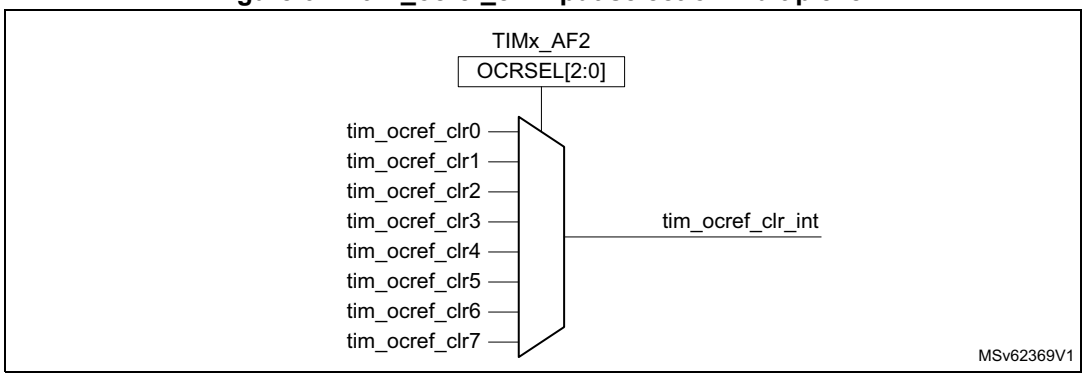


38.4.17 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown in *Figure 641*.

Figure 641. tim_ocref_clr input selection multiplexer



38.4.18 One-pulse mode

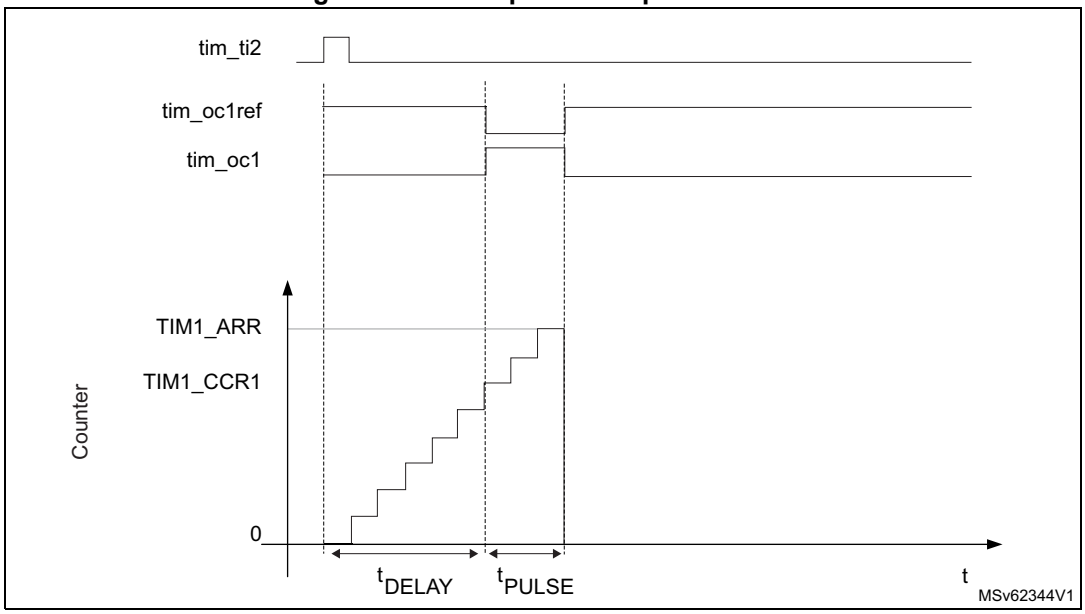
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 642. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[1..15] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS='00110' in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one wants to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to '0' in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input sets the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

38.4.19 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in the *Section 38.4.18: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

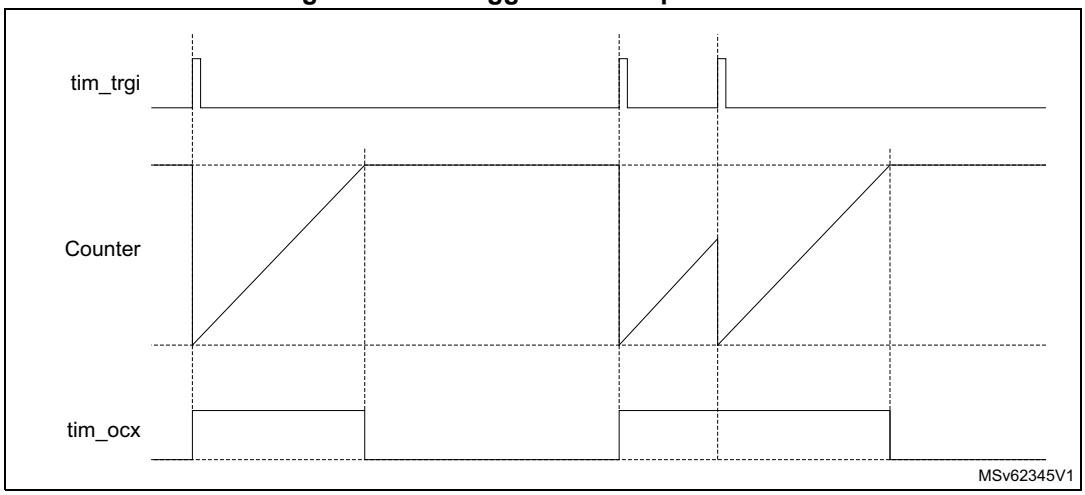
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bits are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 643. Retriggerable one pulse mode



38.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

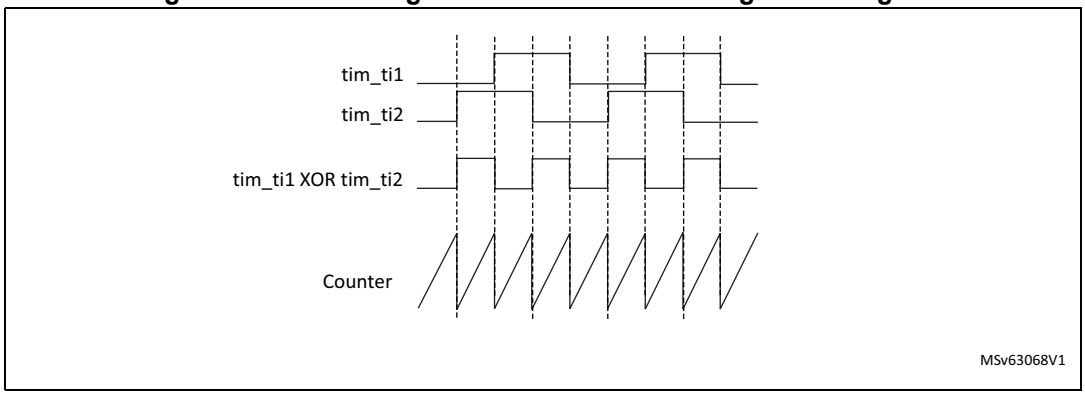
There is no latency between the assertions of the UIF and UIFCPY flags.

38.4.21 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 644*.

Figure 644. Measuring time interval between edges on 2 signals



38.4.22 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

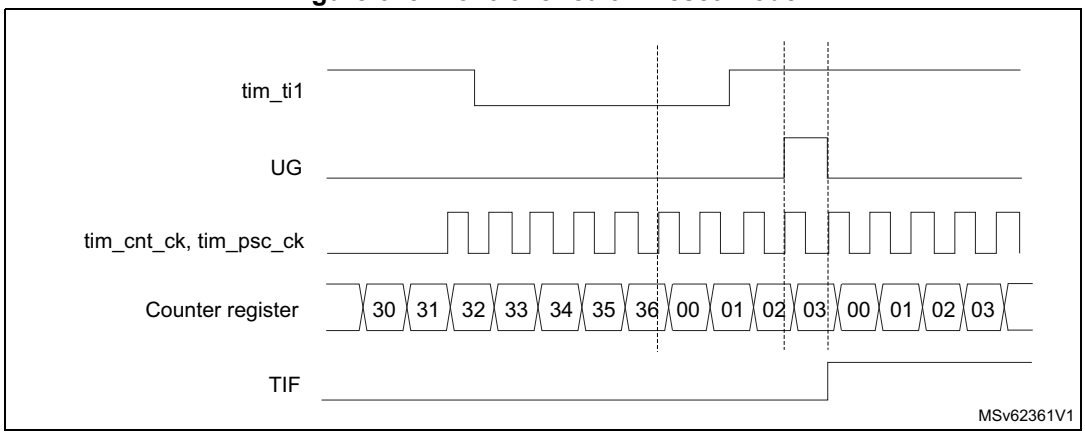
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 645. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

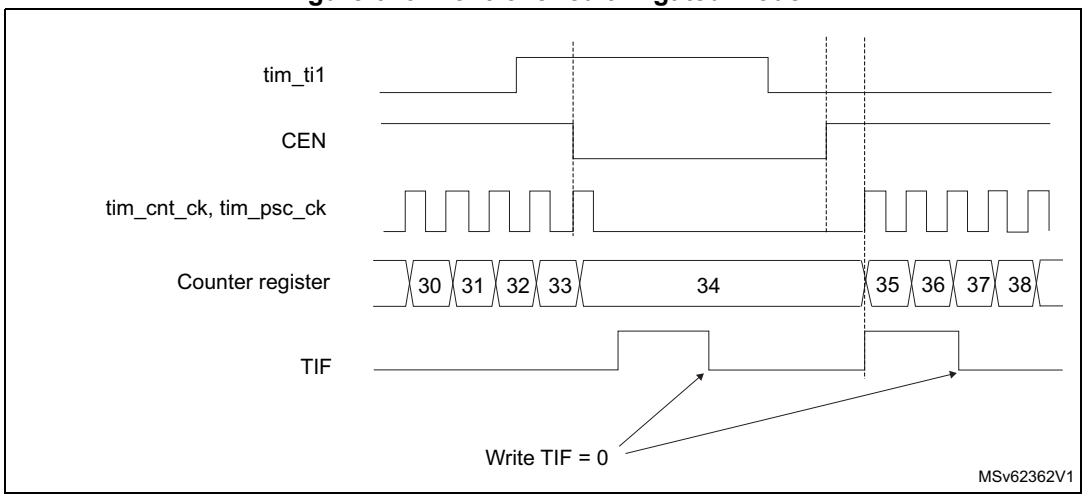
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = '0' in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 646. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

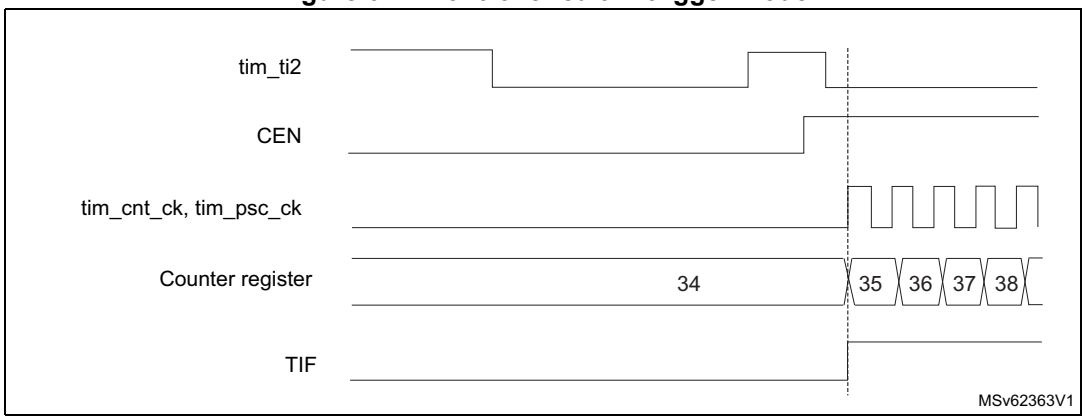
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in the TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 647. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

38.4.23 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

38.4.24 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode allows to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

38.4.25 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 36.4.23: Timer synchronization* for details.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.4.26 Using timer output as trigger for other timers (TIM16)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

38.4.27 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address, that is the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (refer to the note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

38.4.28 TIM15/TIM16 DMA requests

The TIM15/TIM16 can generate a DMA requests, as shown in the *Table 446*.

Table 446. DMA request

DMA acronym	DMA request	Enable control bit
TIM_UP	Update	UDE
TIM_CH1	Capture/compare 1	CC1DE
TIM_CH2(1)	Capture/compare 2	CC2DE
TIM_COM	Commutation (COM)	COMDE
TIM_TRIG(1)	Trigger	TDE

1. Available for TIM15 only.

38.4.29 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

38.5 TIM15/TIM16 low-power modes

Table 447. Effect of low-power modes on TIM15/TIM16

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.

38.6 TIM15/TIM16 interrupts

TIM15/TIM16 can generate multiple interrupts, as shown in *Table 448*.

Table 448. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
TIM	Update	UIF	UIE	Write 0 in UIF	Yes
	Capture/compare 1	CC1IF	CC1IE	Write 0 in CC1IF	Yes
	Capture/compare 2(1)	CC2IF	CC2IE	Write 0 in CC2IF	Yes
	Commutation (COM)	COMIF	COMIE	Write 0 in COMIF	Yes
	Trigger(1)	TIF	TIE	Write 0 in TIF	Yes
	Break	BIF	BIE	Write 0 in BIF	Yes

1. Available for TIM15 only.

38.7 TIM15/TIM16 memory map

Table 449. TIM15/TIM16 register memory map

Offset	Register name
0x00	*TIM15 control register 1 (TIM15_CR1)*
0x04	*TIM15 control register 2 (TIM15_CR2)*
0x08	*TIM15 slave mode control register (TIM15_SMCR)*
0x0C	*TIM15 DMA/interrupt enable register (TIM15_DIER)*
0x10	*TIM15 status register (TIM15_SR)*
0x14	*TIM15 event generation register (TIM15_EGR)*
0x18	*TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)*
0x18	*TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)*
0x20	*TIM15 capture/compare enable register (TIM15_CCER)*

Table 449. TIM15/TIM16 register memory map (continued)

Offset	Register name
0x24	*TIM15 counter (TIM15_CNT)*
0x28	*TIM15 prescaler (TIM15_PSC)*
0x2C	*TIM15 auto-reload register (TIM15_ARR)*
0x30	*TIM15 repetition counter register (TIM15_RCR)*
0x34	*TIM15 capture/compare register 1 (TIM15_CCR1)*
0x38	*TIM15 capture/compare register 2 (TIM15_CCR2)*
0x44	*TIM15 break and dead-time register (TIM15_BDTR)*
0x054	*TIM15 timer deadtime register 2 (TIM15_DTR2)*
0x5C	*TIM15 input selection register (TIM15_TISEL)*
0x060	*TIM15 alternate function register 1 (TIM15_AF1)*
0x064	*TIM15 alternate function register 2 (TIM15_AF2)*
0x3DC	*TIM15 DMA control register (TIM15_DCR)*
0x3E0	*TIM15 DMA address for full transfer (TIM15_DMAR)*
0x00	*TIMx control register 1 (TIMx_CR1)(x = 16)*
0x04	*TIMx control register 2 (TIMx_CR2)(x = 16)*
0x0C	*TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16)*
0x10	*TIMx status register (TIMx_SR)(x = 16)*
0x14	*TIMx event generation register (TIMx_EGR)(x = 16)*
0x18	*TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16)*
0x18	*TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16)*
0x20	*TIMx capture/compare enable register (TIMx_CCER)(x = 16)*
0x24	*TIMx counter (TIMx_CNT)(x = 16)*
0x28	*TIMx prescaler (TIMx_PSC)(x = 16)*
0x2C	*TIMx auto-reload register (TIMx_ARR)(x = 16)*
0x30	*TIMx repetition counter register (TIMx_RCR)(x = 16)*
0x34	*TIMx capture/compare register 1 (TIMx_CCR1)(x = 16)*
0x44	*TIMx break and dead-time register (TIMx_BDTR)(x = 16)*
0x054	*TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16)*
0x5C	*TIMx input selection register (TIMx_TISEL)(x = 16)*
0x060	*TIMx alternate function register 1 (TIMx_AF1)(x = 16)*
0x064	*TIMx alternate function register 2 (TIMx_AF2)(x = 16)*
0x3DC	*TIMx DMA control register (TIMx_DCR)(x = 16)*
0x3E0	*TIM16 DMA address for full transfer (TIMx_DMAR)(x = 16)*

38.8 TIM15 registers

38.8.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM15_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM15_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit field indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM15_ARR register is not buffered
1: TIM15_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

38.8.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)

0: tim_oc2=0 when MOE=0

1: tim_oc2=1 when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n=0 after a dead-time when MOE=0

1: tim_oc1n=1 after a dead-time when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1=0 after a dead-time when MOE=0

1: tim_oc1=1 after a dead-time when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15..0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15..0] and tim_ti2_in[15..0] multiplexer outputs are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: **Reset** - the UG bit from the TIM15_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM15_SMCR register).

010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

38.8.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bit field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
Others: Reserved

Refer to *Section 38.4.2: TIM15/TIM16 pins and internal signals* for more details on tim_itrx meaning for each timer.

Note: These bits must be changed only when they are not used (for example, when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=’00100’). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC...) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.8.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

38.8.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected

1: The counter value has been captured in TIM15_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred

1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match.

1: The content of the counter TIM15_CNT matches the content of the TIM15_CCR1 register. When the contents of TIM15_CCR1 are greater than the contents of TIM15_ARR, the CC1IF bit goes high on the counter overflow.

If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIM15_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIM15_CCR1 register (An edge has been detected on tim_ic1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIM15_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIM15_EGR register, if URS=0 and UDIS=0 in the TIM15_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 38.8.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS=0 and UDIS=0 in the TIM15_CR1 register.

38.8.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM15_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1 A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM15_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

38.8.7 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIM15_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit field defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E='0' (TIM15_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIM15_CCER).

38.8.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIM15_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIM15_CCR1 and the counter TIM15_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM15_CNT=TIM15_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIM15_CNT<TIM15_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIM15_CNT<TIM15_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channel becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channel becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channel becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channel becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved,

*Note: **1:** These bits cannot be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** In PWM mode, the tim_ocxref level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

***3**: On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM15_CCR1 disabled. TIM15_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM15_CCR1 enabled. Read/Write operations access the preload register. TIM15_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits cannot be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIM15_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIM15_CCER).

38.8.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: 1.This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S="00" (the channel is configured in output).

2. On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: tim_oc1 active high

1: tim_oc1 active low

CC1 channel configured as input: The CC1NP/CC1P bits select the polarity of tim_ti1fp1 and tim_ti2fp1 for trigger or capture operations.

00: non-inverted/rising edge. The circuit is sensitive to tim_tixfp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

01: inverted/falling edge. The circuit is sensitive to tim_tixfp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is inverted (trigger operation in gated mode).

10: Reserved

11: non-inverted/both edges. The circuit is sensitive to both tim_tixfp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

Note: 1. This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

2. On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:

0: Off - tim_oc1 is not active. tim_oc1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - tim_oc1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIM15_CCR1) or not.

0: Capture disabled

1: Capture enabled

Table 450. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_ocx=0 tim_ocxn=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_ocx=0	tim_ocxref + Polarity tim_ocxn=tim_ocxref XOR CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn=0
		X	1	1	tim_ocxref + Polarity + dead-time	Complementary to tim_ocxref (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx=CCxP	tim_ocxref + Polarity tim_ocxn=tim_ocxref XOR CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn=CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx=CCxP, tim_ocxn=CCxNP Then if the clock is present: tim_ocx=OISx and tim_ocxn=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_ocxn both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and AFIO registers.

38.8.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM15_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

38.8.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM15_EGR register or through trigger controller when configured in “reset mode”).

38.8.12 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to *Section 38.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bit field contains the dithered part.

38.8.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bit field defines the update rate of the compare registers (that are periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM15_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

- the number of PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode

38.8.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bit field contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

38.8.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bit field contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

38.8.16 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM15_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bit field defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM15_CCER register)

Refer to tim_ocx/tim_ocxn enable description for more details (*Section 38.8.9: TIM15 capture/compare enable register (TIM15_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

*Note: **1:** This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

***2:** Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled

1; Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

Refer to tim_ocx/tim_ocxn enable description for more details (*Section 38.8.9: TIM15 capture/compare enable register (TIM15_CCER)*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)

1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit cannot be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

Refer to tim_ocx/tim_ocxn enable description for more details (*Section 38.8.9: TIM15 capture/compare enable register (TIM15_CCER)*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal=0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. tim_ocx/tim_ocxn enable output signal=1)

Note: This bit cannot be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM15_BDTR register, OISx and OISxN bits in TIM15_CR2 register and BKBID/BKE/BKP/AOE bits in TIM15_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM15_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM15_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM15_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit field defines the duration of the dead-time inserted between the complementary outputs. DT corresponds to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$

DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$

DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$

DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bit field cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

38.8.17 TIM15 timer deadtime register 2 (TIM15_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.

DTGF[7:5]=10x => DTF=(64+DTGF[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.

DTGF[7:5]=110 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.

DTGF[7:5]=111 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

Note: This bit field cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

38.8.18 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[0..15] input

0000: TIM15_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

1111: tim_ti2_in15

Refer to *Section 38.4.2: TIM15/TIM16 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[0..15] input

0000: TIM15_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Section 38.4.2: TIM15/TIM16 pins and internal signals* for interconnects list.

38.8.19 TIM15 alternate function register 1 (TIM15_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 38.4.2: TIM15/TIM16 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. mdf_brkx output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. COMP7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

38.8.20 TIM15 alternate function register 2 (TIM15_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 38.4.2: TIM15/TIM16 pins and internal signals* for product specific implementation.

Note: These bits cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

38.8.21 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM15_DMAR address).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIM15_DMAR address). DBA is defined as an offset starting from the address of the TIM15_CR1 register.
Example:
00000: TIM15_CR1,
00001: TIM15_CR2,
00010: TIM15_SMCR,
...

38.8.22 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM15_CR1 address) + (DBA + DMA index) x 4

where TIM15_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM15_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM15_DCR).

38.9 TIM16 registers

38.9.1 TIMx control register 1 (TIMx_CR1)(x = 16)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit field indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),

00: $t_{DTS}=t_{tim_ker_ck}$

01: $t_{DTS}=2*t_{tim_ker_ck}$

10: $t_{DTS}=4*t_{tim_ker_ck}$

11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller

1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

38.9.2 TIMx control register 2 (TIMx_CR2)(x = 16)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n=0 after a dead-time when MOE=0
1: tim_oc1n=1 after a dead-time when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1=0 after a dead-time when MOE=0
1: tim_oc1=1 after a dead-time when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

38.9.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
		rw				rw	rw	rw		rw				rw	rw

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

38.9.4 TIMx status register (TIMx_SR)(x = 16)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag
If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on tim_ic1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

38.9.5 TIMx event generation register (TIMx_EGR)(x = 16)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

38.9.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit field defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

38.9.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example, channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

Others: Reserved

*Note: **1:** These bits cannot be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** In PWM mode 1 or 2, the tim_oc1ref level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits cannot be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

38.9.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

*Note: **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).*

***2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.*

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: tim_oc1 active high

1: tim_oc1 active low

CC1 channel configured as input:

The CC1NP/CC1P bits select the polarity of tim_ti1fp1 for trigger or capture operations.

00: Non-inverted/rising edge. The circuit is sensitive to tim_ti1fp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is not inverted (trigger operation in gated mode).

01: Inverted/falling edge. The circuit is sensitive to tim_ti1fp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is inverted (trigger operation in gated mode.

10: Reserved

11: Non-inverted/both edges. The circuit is sensitive to both tim_ti1fp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is not inverted (trigger operation in gated mode).

*Note: **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

***2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:

0: Off - tim_oc1 is not active. tim_oc1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - tim_oc1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled

1: Capture enabled

Table 451. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CC1E bit	CC1NE bit	tim_oc1 output state	tim_oc1n output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_oc1=0 tim_oc1n=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_oc1=0	tim_oc1ref + Polarity tim_oc1n=tim_oc1ref XOR CC1NP
		0	1	0	tim_oc1ref + Polarity tim_oc1=tim_oc1ref XOR CC1P	Output Disabled (not driven by the timer: Hi-Z) tim_oc1n=0
		X	1	1	tim_oc1ref + Polarity + dead-time	Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_oc1=CC1P	tim_oc1ref + Polarity tim_oc1n=tim_oc1ref XOR CC1NP
		1	1	0	tim_oc1ref + Polarity tim_oc1=tim_oc1ref XOR CC1P	Off-State (output enabled with inactive state) tim_oc1n=CC1NP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_oc1=CC1P, tim_oc1n=CC1NP Then if the clock is present: tim_oc1=OIS1 and tim_oc1n=OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and AFIO registers.

38.9.9 TIMx counter (TIMx_CNT)(x = 16)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

38.9.10 TIMx prescaler (TIMx_PSC)(x = 16)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

38.9.11 TIMx auto-reload register (TIMx_ARR)(x = 16)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to *Section 38.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bit field contains the dithered part.

38.9.12 TIMx repetition counter register (TIMx_RCR)(x = 16)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bit field defines the update rate of the compare registers (that are periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

- the number of PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode

38.9.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bit field contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

38.9.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bit field defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING}=f_{tim_ker_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim_ker_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim_ker_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

Refer to tim_oc1/tim_oc1n enable description for more details (*Section 38.9.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

*Note: **1.** This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

***2.** Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

*Note: **1.** This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

***2.** Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

Refer to tim_oc1/tim_oc1n enable description for more details (*Section 38.9.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E=1 or CC1NE=1 (the output is still controlled by the timer).

Note: This bit cannot be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

Refer to tim_oc1/tim_oc1n enable description for more details (*Section 38.9.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal=0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E=1 or CC1NE=1. tim_oc1/tim_oc1n enable output signal=1)

Note: This bit cannot be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit field defines the duration of the dead-time inserted between the complementary outputs. DT corresponds to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$

DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$

DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$

DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bit field cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

38.9.15 TIMx option register 1 (TIMx_OR1)(x = 16)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE32EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **HSE32EN**: HSE Divided by 32 enable

This bit enables the HSE divider by 32 for the tim_ti1_in3. Refer to *Table 59: Interconnect to the TIMx tim_ti1 input multiplexer (x=15, 16)* for details.

0: HSE divided by 32 disabled

1: HSE divided by 32 enabled

38.9.16 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.

DTGF[7:5]=10x => DTF=(64+DTGF[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.

DTGF[7:5]=110 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.

DTGF[7:5]=111 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

Note: This bit field cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

38.9.17 TIMx input selection register (TIMx_TISEL)(x = 16)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[0..15] input

0000: TIMx_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Section 38.4.2: TIM15/TIM16 pins and internal signals* for interconnects list.

38.9.18 TIMx alternate function register 1 (TIMx_AF1)(x = 16)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 38.4.2: TIM15/TIM16 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. mdf_brkx output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

38.9.19 TIMx alternate function register 2 (TIMx_AF2)(x = 16)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection

These bits select the tim_ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 38.4.2: TIM15/TIM16 pins and internal signals* for product specific implementation.

Note: These bits cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

38.9.20 TIMx DMA control register (TIMx_DCR)(x = 16)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), that is the number of transfers. Transfers can be in half-words or in bytes (refer to example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

38.9.21 TIM16 DMA address for full transfer (TIMx_DMAR)(x = 16)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

39 Timestamp timer (TIM_TS)

39.1 TIM_TS introduction

The timestamp timer TIM_TS consists in a 32-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation.

The timers are completely independent, and do not share any resources.

39.2 TIM_TS main features

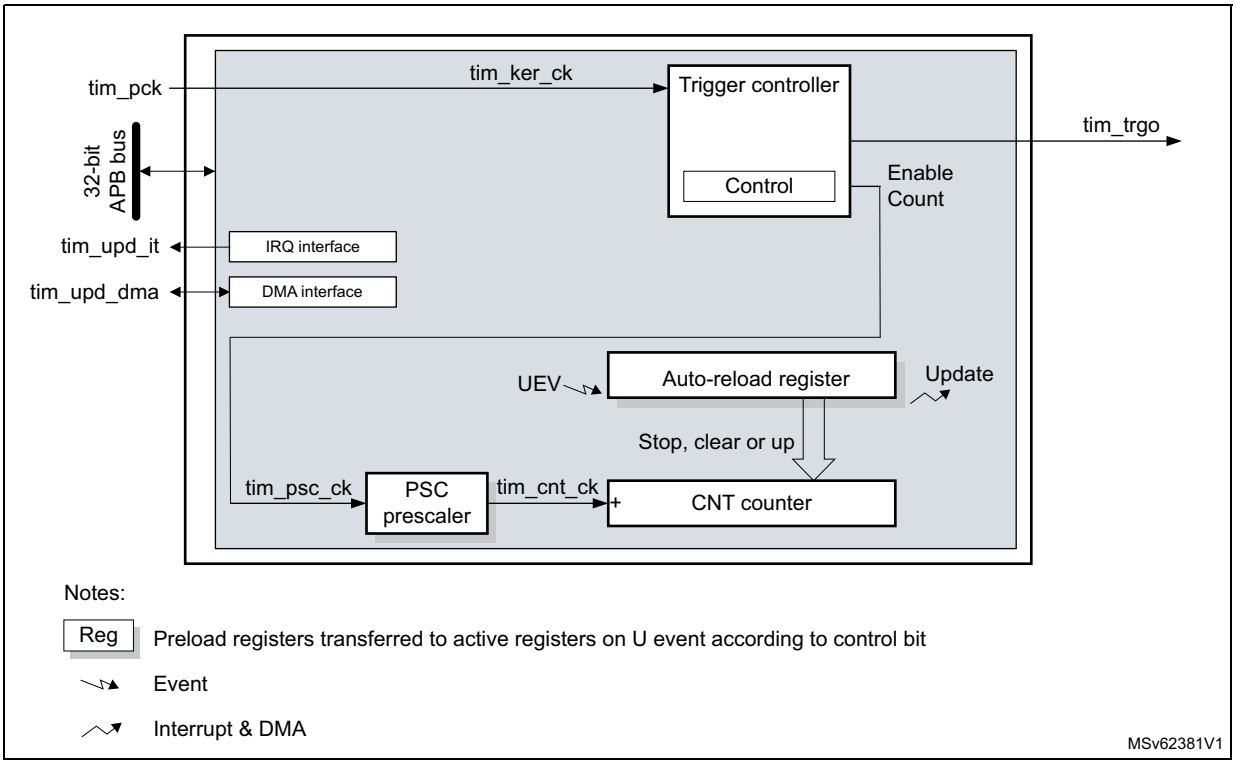
Timestamp timer (TIM_TS) features include:

- 32-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Interrupt/DMA generation on the update event: counter overflow

39.3 TIM_TS functional description

39.3.1 TIM_TS block diagram

Figure 648. Timestamp timer block diagram



39.3.2 TIM_TS internal signals

The below table summarizes the TIM inputs and outputs.

Table 452. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

39.3.3 TIM_TS clocks

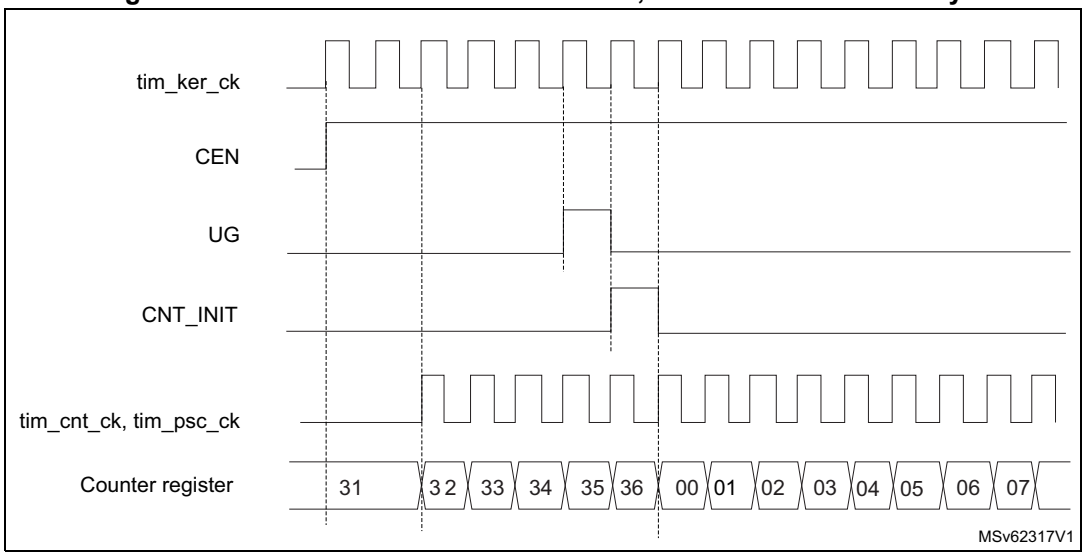
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIM_TS_CR1 register) and UG bits (in the TIM_TS_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 649 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 649. Control circuit in normal mode, internal clock divided by 1



39.3.4 Time-base unit

The main block of the programmable timer is a 32-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIM_TS_CNT)
- Prescaler Register (TIM_TS_PSC)
- Auto-Reload Register (TIM_TS_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIM_TS_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIM_TS_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIM_TS_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set 1 clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_TS_PSC register). It can be changed on the fly as the TIM_TS_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 650 and *Figure 651* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 650. Counter timing diagram with prescaler division change from 1 to 2

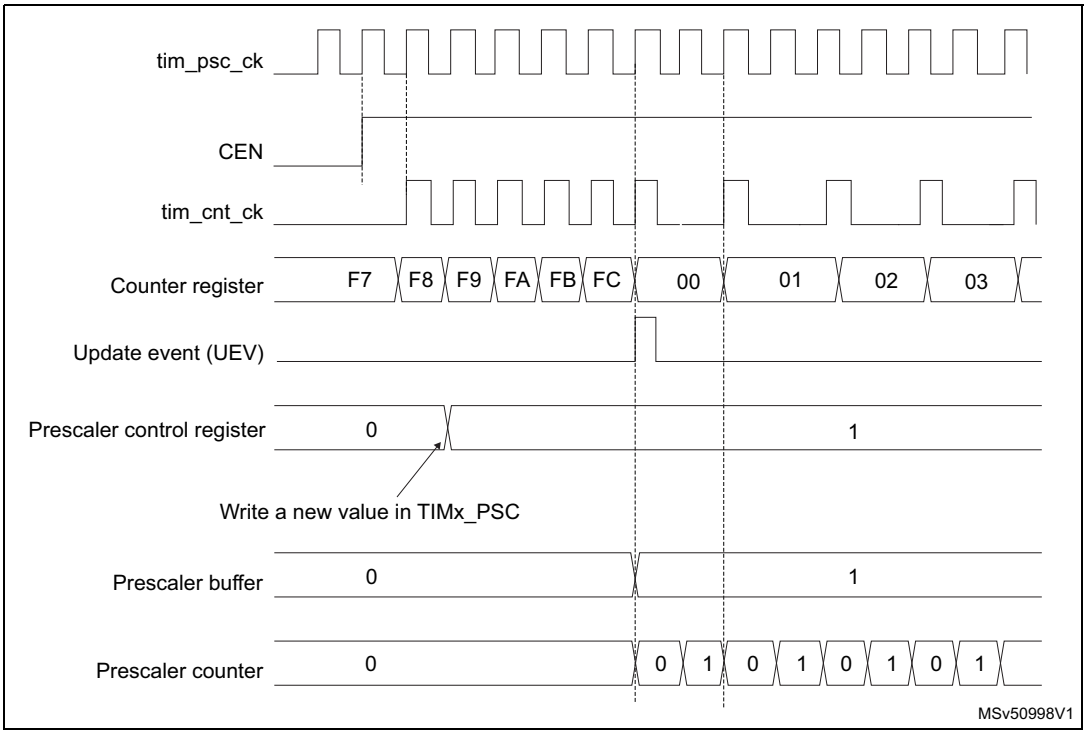
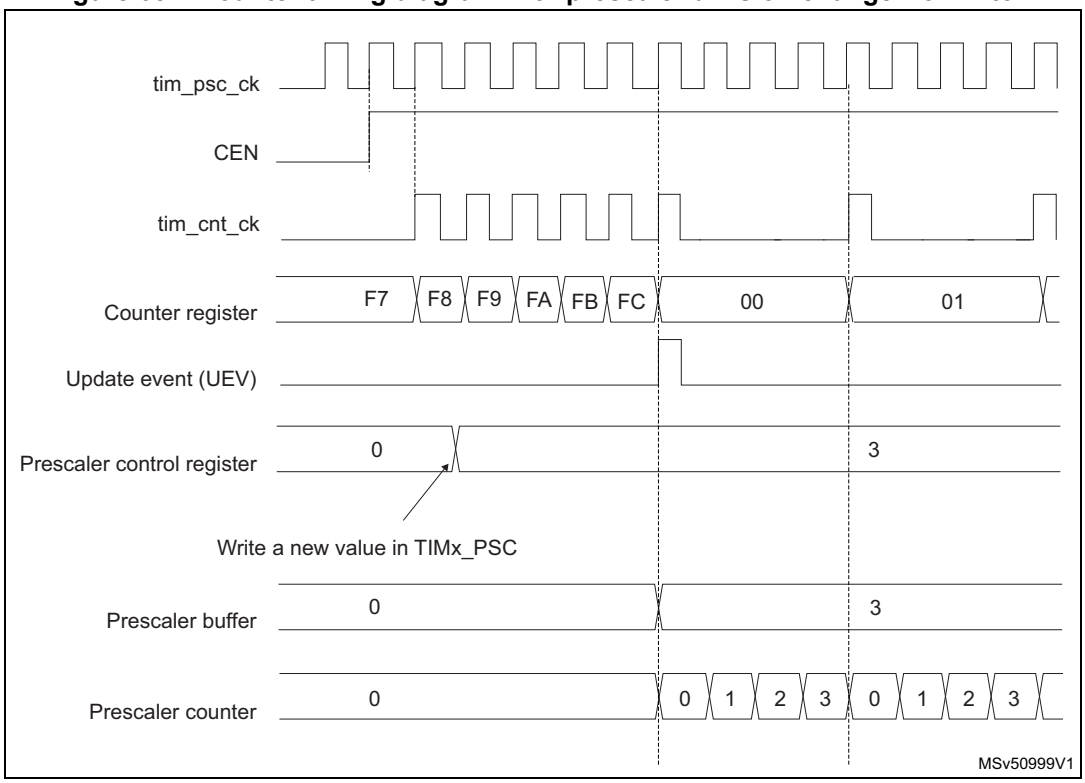


Figure 651. Counter timing diagram with prescaler division change from 1 to 4



39.3.5 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIM_TS_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIM_TS_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIM_TS_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIM_TS_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIM_TS_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIM_TS_PSC register)
- The auto-reload shadow register is updated with the preload value (TIM_TS_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIM_TS_ARR = 0x36.

Figure 652. Counter timing diagram, internal clock divided by 1

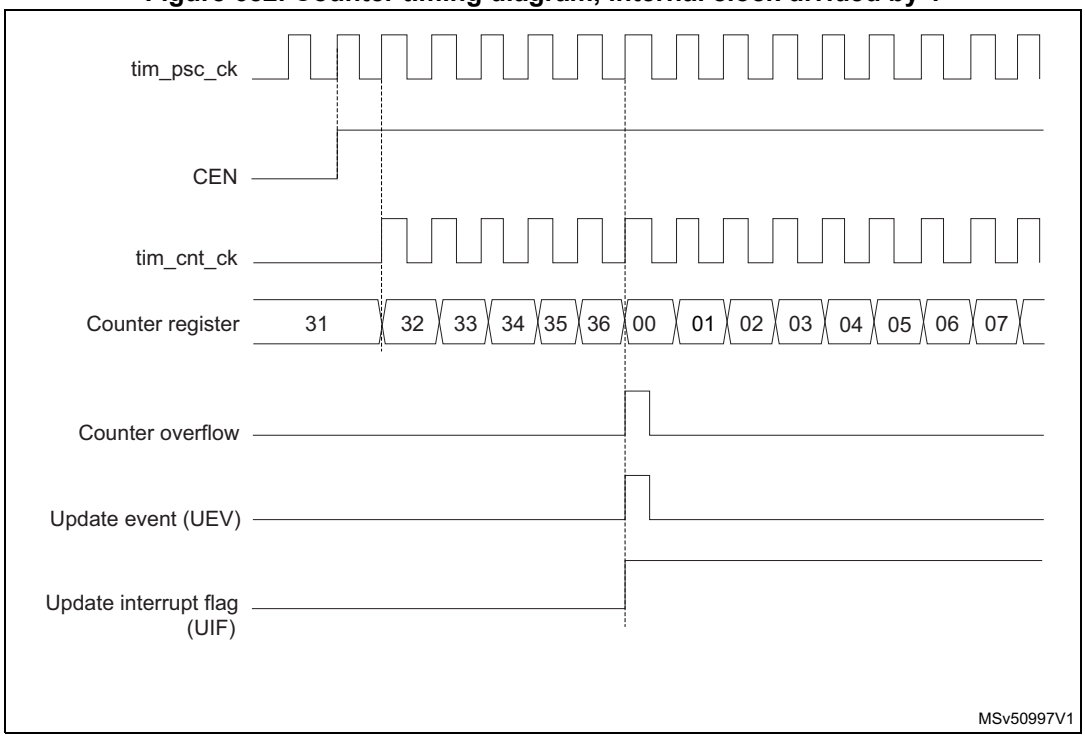


Figure 653. Counter timing diagram, internal clock divided by 2

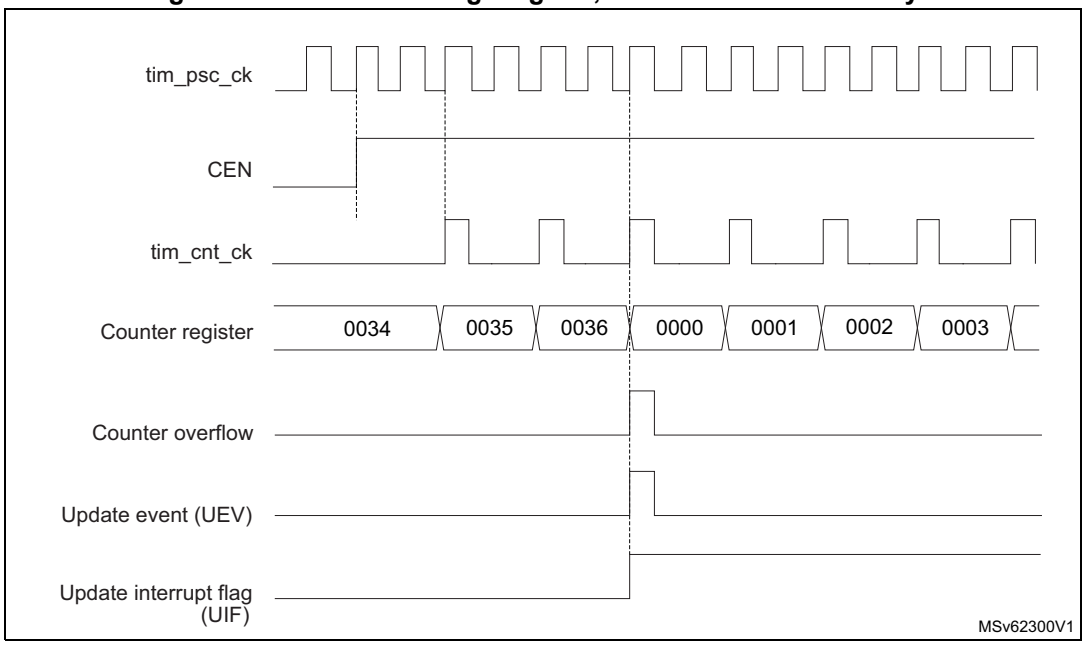


Figure 654. Counter timing diagram, internal clock divided by 4

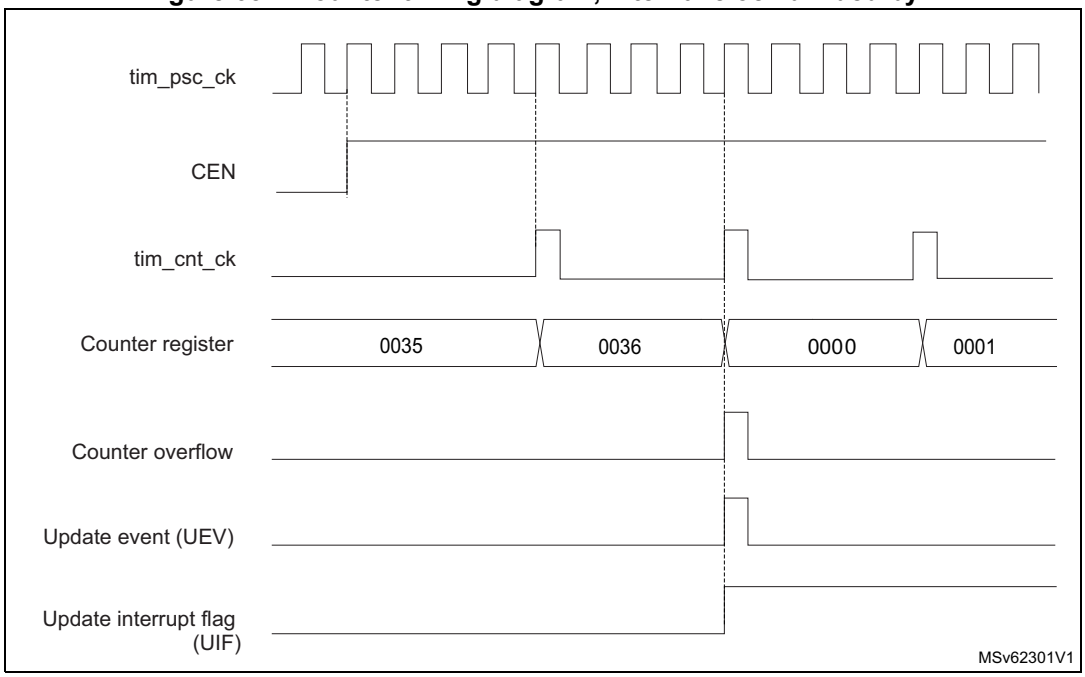


Figure 655. Counter timing diagram, internal clock divided by N

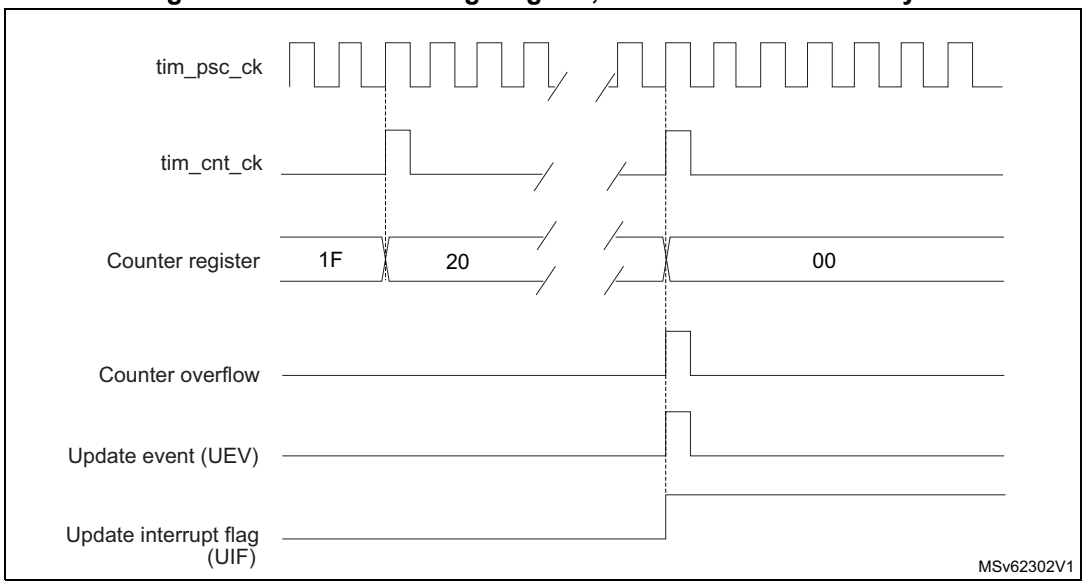


Figure 656. Counter timing diagram, update event when ARPE = 0 (TIM_TS_ARR not preloaded)

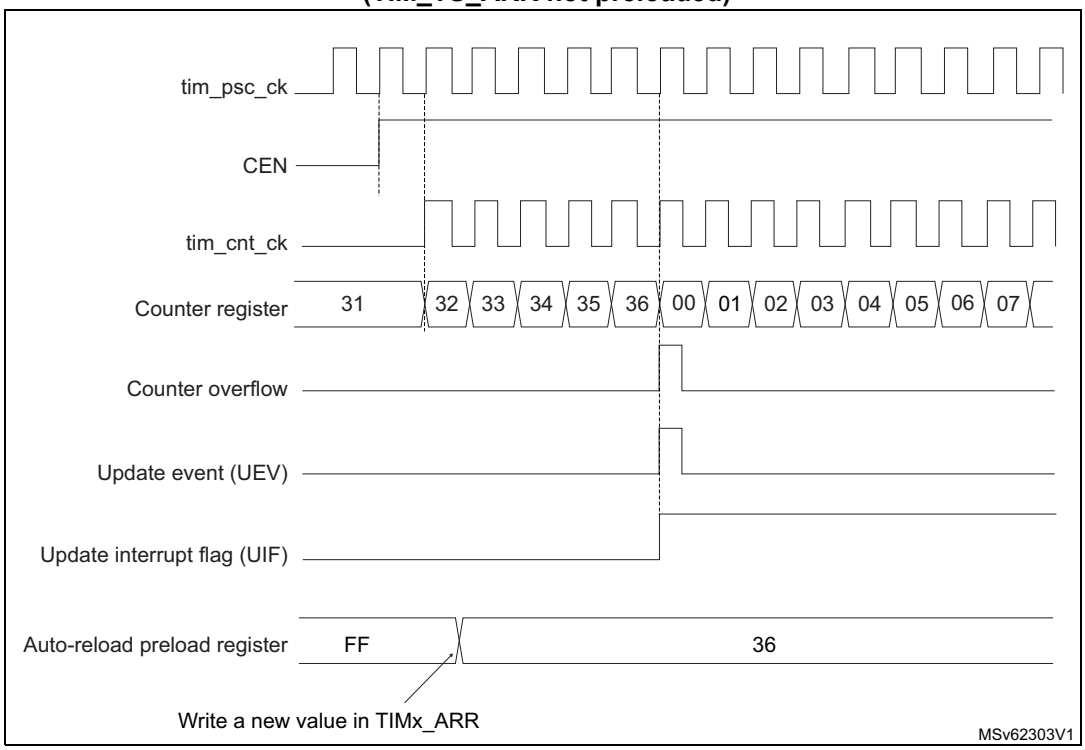
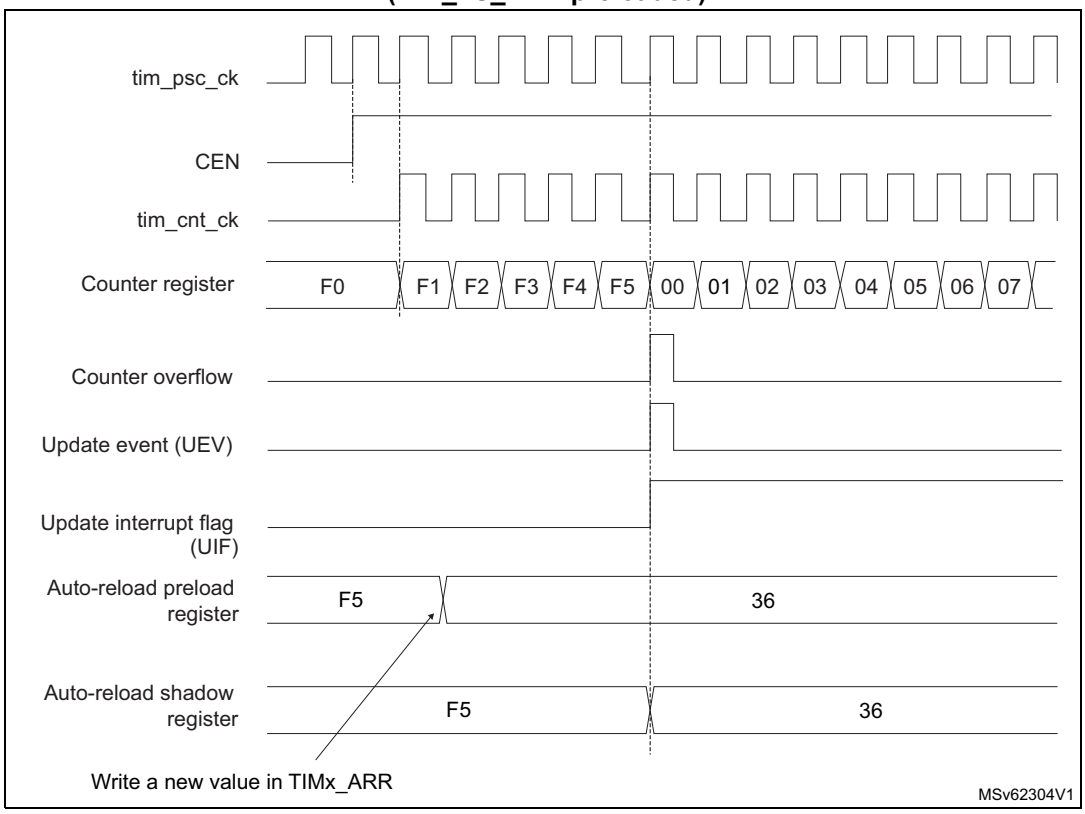


Figure 657. Counter timing diagram, update event when ARPE=1 (TIM_TS_ARR preloaded)



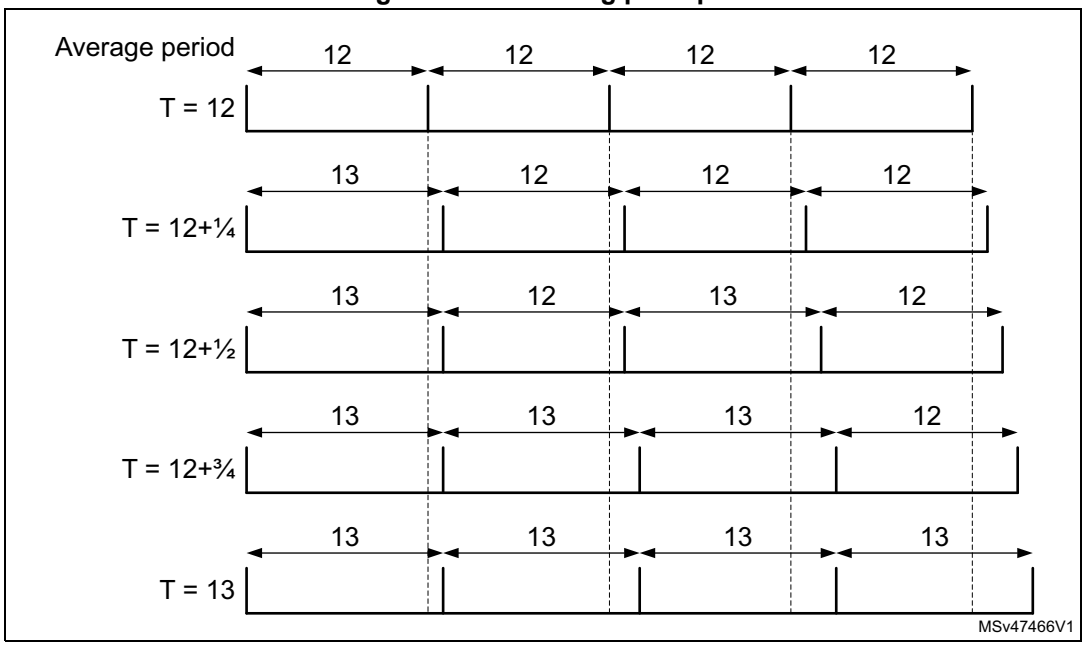
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_TS_CR1 register. This affects the way the TIM_TS_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

Figure 658 below presents the dithering principle applied to 4 consecutive counting periods.

Figure 658. Dithering principle



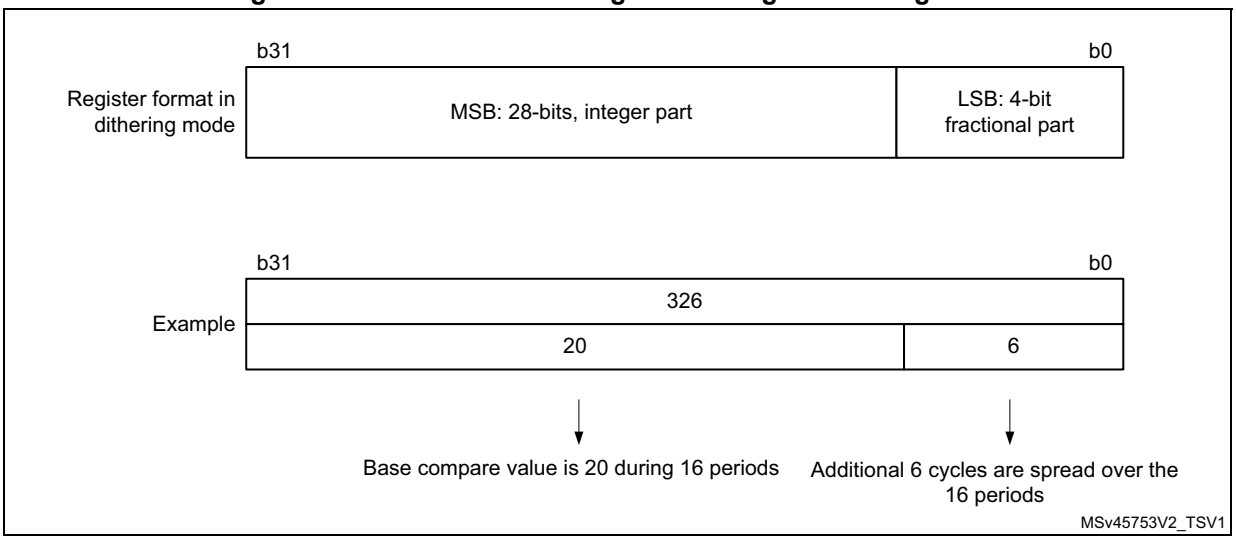
When the dithering mode is enabled, the register coding is changed as follows (see *Figure 659* for example):

- The 4 LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The ARR values are updated automatically if the DITHEN bit is set / reset (for instance, if ARR= 0x05 with DITHEN = 0, it is updated to ARR = 0x50 with DITHEN = 1).The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset*
2. *The ARR[3:0] bits must be reset*
3. *The DITHEN bit must be reset*
4. *The CEN bit can be set (eventually with ARPE = 1)*

Figure 659. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{Cnt}} \Rightarrow F_{CntMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

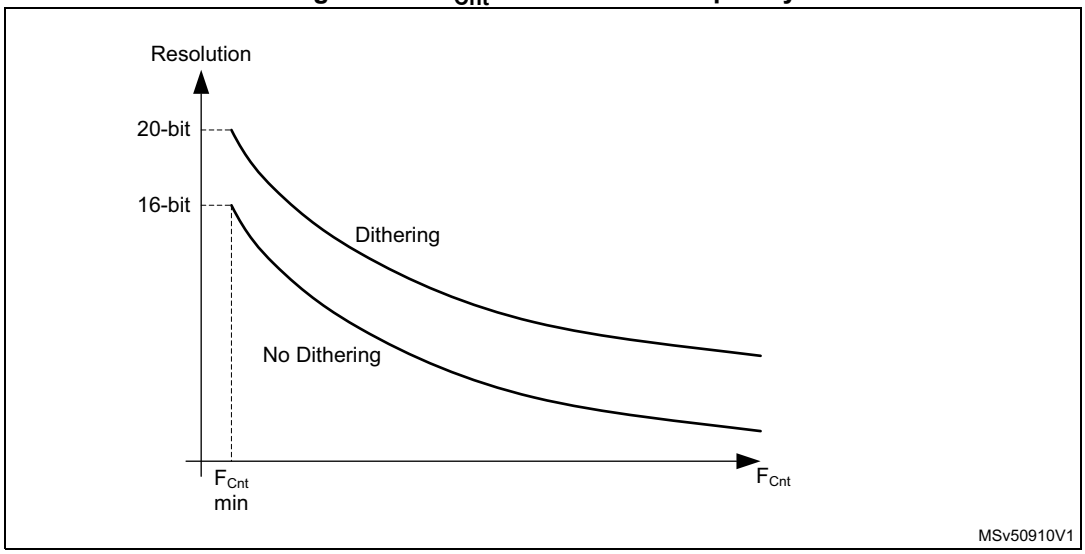
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_TS_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

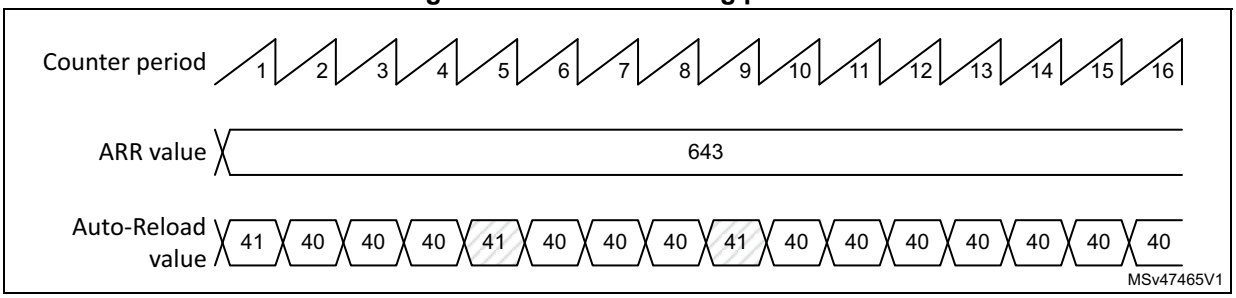
As shown in *Figure 660* below, the dithering mode allows to increase the PWM resolution whatever the PWM frequency.

Figure 660. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 661* below.

Figure 661. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in *Table 453* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 453. TIM_TS_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

39.3.6 UIF bit remapping

The IUFREMAP bit in the TIM_TS_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIM_TS_CNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

39.3.7 TIM_TS DMA requests

The TIM_TS can generate a single DMA request, as shown in *Table 454*.

Table 454. DMA request

DMA acronym	DMA request	Enable control bit
TIM_TS_UP	Update	UDE

39.3.8 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 with FPU core halted), the TIM_TS counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

39.3.9 TIM_TS low-power modes

Table 455. Effect of low-power modes on TIM_TS

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.

39.3.10 TIM_TS interrupts

The TIM_TS can generate a single interrupt, as shown in *Table 456*.

Table 456. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
TIM_TS	Update	UIF	UIE	write 0 in UIF	Yes

39.4 TIM_TS registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

39.4.1 TIM_TS memory map

Table 457. TIM_TS register memory map

Offset	Register name
0x00	*TIM_TS control register 1 (TIM_TS_CR1)*
0x04	*TIM_TS control register 2 (TIM_TS_CR2)*
0x0C	*TIM_TS DMA/Interrupt enable register (TIM_TS_DIER)*
0x10	*TIM_TS status register (TIM_TS_SR)*
0x14	*TIM_TS event generation register (TIM_TS_EGR)*
0x24	*TIM_TS counter (TIM_TS_CNT)*
0x28	*TIM_TS prescaler (TIM_TS_PSC)*
0x2C	*TIM_TS auto-reload register (TIM_TS_ARR)*

39.4.2 TIM_TS control register 1 (TIM_TS_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_TS_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIM_TS_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_TS_ARR register is not buffered.

1: TIM_TS_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.
CEN is cleared automatically in one-pulse mode, when an update event occurs.

39.4.3 TIM_TS control register 2 (TIM_TS_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIM_TS_EGR register is used as a trigger output (tim_trgo).
001: **Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.
010: **Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.
Note: The clock of the slave timer or the peripheral receiving the tim_trgo must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

39.4.4 TIM_TS DMA/Interrupt enable register (TIM_TS_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

39.4.5 TIM_TS status register (TIM_TS_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– On counter overflow if UDIS = 0 in the TIM_TS_CR1 register.
– When CNT is reinitialized by software using the UG bit in the TIM_TS_EGR register, if URS = 0 and UDIS = 0 in the TIM_TS_CR1 register.

39.4.6 TIM_TS event generation register (TIM_TS_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

39.4.7 TIM_TS counter (TIM_TS_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31]/ UIFCPY	CNT[30:16]														
rw/r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CNT[31] or UIFCPY**: Value depends on IUFREMAP in TIM_TS_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIM_TS_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0):
The register holds the counter value.
Dithering mode (DITHEN = 1):
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

39.4.8 TIM_TS prescaler (TIM_TS_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency $f_{tim_cnt_ck}$ is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_TS_EGR register).

39.4.9 TIM_TS auto-reload register (TIM_TS_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARR[31:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 39.3.4: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bit field contains the dithered part.

40 Real-time clock (RTC)

40.1 Introduction

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, sleep mode).

40.2 RTC main features

The RTC supports the following features (see *Figure 662: RTC block diagram*):

- Calendar with subsecond, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- One programmable alarm.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- 17-bit auto-reload wakeup timer (WUT) for periodic events with programmable resolution and period.

The RTC is supplied through a switch that takes power from the V_{DD} supply.

The RTC clock sources can be:

- The internal low power RC oscillator (LSI, with typical frequency of 1024 kHz)
- The high-speed external clock (HSE), divided by a prescaler in the RCC.

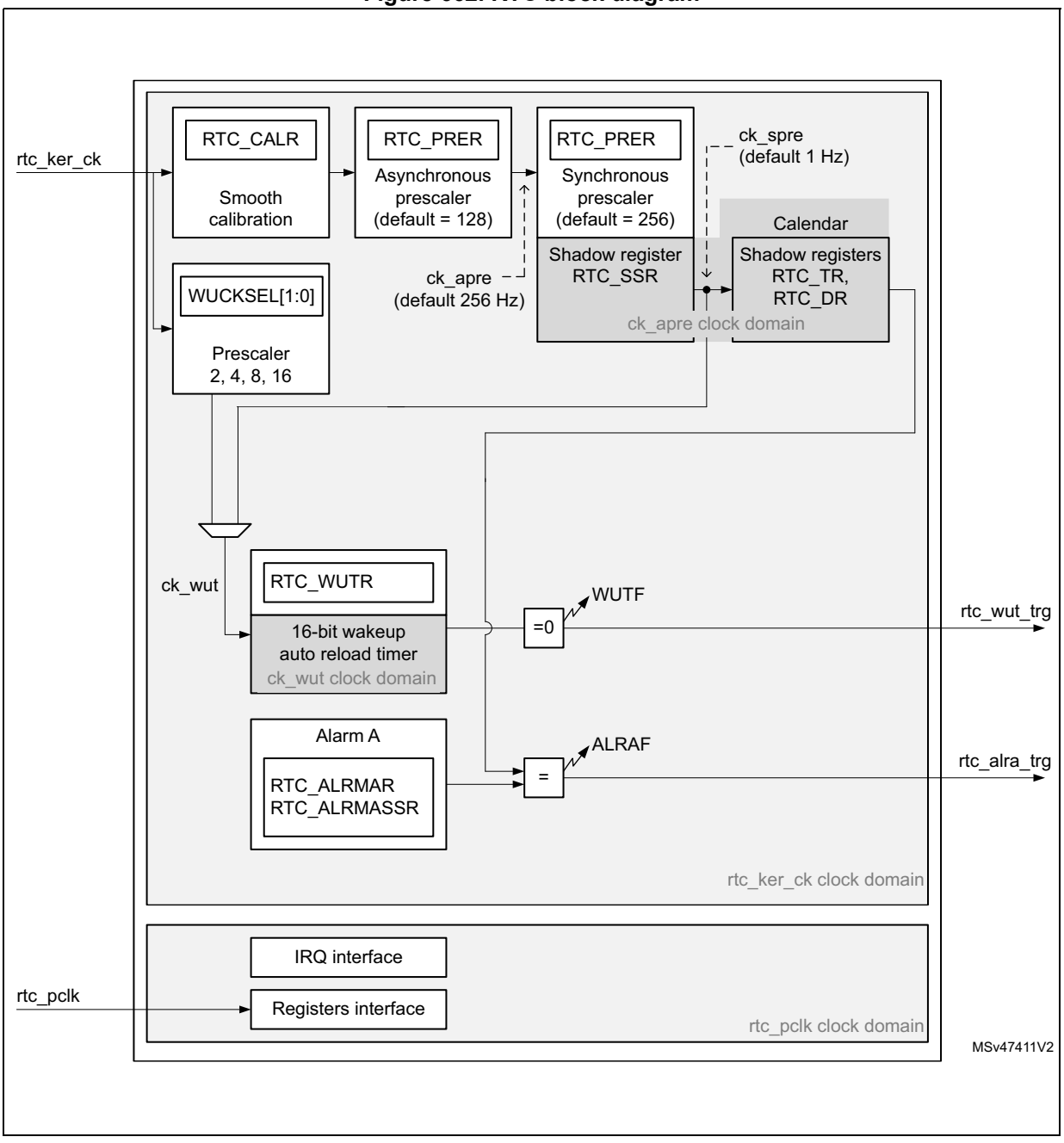
When clocked by the LSI, the RTC is functional in all low-power modes.

All RTC events (Alarm and WakeUp Timer) can generate an interrupt and wakeup the device from the low-power modes. Refer to *Table 17: EXTI event input mapping* for more details.

40.3 RTC functional description

40.3.1 RTC block diagram

Figure 662. RTC block diagram



40.3.2 RTC pins and internal signals

Table 458. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_it	Output	RTC interrupts (refer to *Section 40.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_wut_trg	Output	RTC wakeup timer event detection trigger

The RTC kernel clock is usually the LSI at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes when the selected clock is not LSI. Refer to *Section 40.4: RTC low-power modes* for more details.

40.3.3 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to the chapter Reset and clock control (RCC).

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 662: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSI frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 40.3.6: Periodic auto-wakeup* for details).

40.3.4 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 40.6.11: RTC shift control register (RTC_SHIFTR)*).

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

40.3.5 Programmable alarm

The RTC unit provides one programmable alarm enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

40.3.6 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSI (32.768 kHz), this allows to configure the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock)

  When ck_spre frequency is 1 Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wakeup timer*), the timer starts counting down. When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

System reset, as well as sleep modes have no influence on the wakeup timer.

40.3.7 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD = 0.

RTC register write protection

Writing to the protected RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarm.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again.

The wakeup timer restarts down-counting.The WUTWF bit is cleared up to 2 RTCCLK clocks cycles after WUTE is cleared, due to clock synchronization.

40.3.8 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode, RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 40.3.10: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes, since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

40.3.9 Resetting the RTC

When a reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

40.3.10 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

RTC_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]. The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (PREDIV_S + 1) seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock.

Caution: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow occurs.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

40.3.11 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Calibration when PREDIV_A < 3

The CALP bit cannot be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP is already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) has to be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S has to be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S has to be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the following process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1.
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

40.3.12 Debug mode

When the device enters debug mode (core halted), the RTC counter either continues to work normally or stops, depending on the configuration of the corresponding bit in DBGMCU freeze registers.

40.4 RTC low-power modes

Table 459. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the sleep mode.

40.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

Table 460. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from Sleep mode
RTC	Alarm A	ALRAF	ALRAIE	Write 1 in CALRAF	Yes
	Wakeup timer interrupt	WUTF	WUTIE	Write 1 in CWUTF	Yes

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.

40.6 RTC registers

Refer to *Section 1.2: Register conventions* of the Preface chapter for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

40.6.1 RTC memory map

Table 461. RTC register memory map

Offset	Register name
0x00	*RTC time register (RTC_TR)*
0x04	*RTC date register (RTC_DR)*
0x08	*RTC sub second register (RTC_SSR)*
0x0C	*RTC initialization control and status register (RTC_ICSR)*
0x10	*RTC prescaler register (RTC_PRER)*
0x14	*RTC wakeup timer register (RTC_WUTR)*
0x18	*RTC control register (RTC_CR)*
0x24	*RTC write protection register (RTC_WPR)*
0x28	*RTC calibration register (RTC_CALR)*

Table 461. RTC register memory map (continued)

Offset	Register name
0x2C	*RTC shift control register (RTC_SHIFTR)*
0x40	*RTC alarm A register (RTC_ALRMAR)*
0x44	*RTC alarm A sub second register (RTC_ALRMASSR)*
0x50	*RTC status register (RTC_SR)*
0x54	*RTC masked interrupt status register (RTC_MISR)*
0x5C	*RTC status clear register (RTC_SCR)*

40.6.2 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration* and *Section 40.3.8: Reading the calendar*.

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

40.6.3 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration* and *Section 40.3.8: Reading the calendar*.

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x04

Reset value: 0x0000 2101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

40.6.4 RTC sub second register (RTC_SSR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SS[15:0]**: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

40.6.5 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x0C

Reset value: 0x0000 0007

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	ALRAWF
								rw	r	rc_w0	r	r	r		r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Wakeup timer configuration update not allowed except in initialization mode

1: Wakeup timer configuration update allowed

Bit 0 **ALRAWF**: Alarm A write flag

This bit is set by hardware when alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm A update not allowed

1: Alarm A update allowed

40.6.6 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration*.

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x10

Reset value: 0x007F 00FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

40.6.7 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x14

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

40.6.8 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKP	SUB1H	ADD1H
													rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WUTIE	Res.	ALRAIE	Res.	WUTE	Res.	ALRAE	Res.	FMT	BYPSHAD	REFCKON	Res.	WUCKSEL[2:0]		
	rw		rw		rw		rw		rw	rw	rw		rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WUTIE**: Wakeup timer interrupt enable

0: Wakeup timer interrupt disabled

1: Wakeup timer interrupt enabled

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled

1: Alarm A interrupt enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **WUTE**: Wakeup timer enable

0: Wakeup timer disabled

1: Wakeup timer enabled

Note: When the wakeup timer is disabled, wait for WUTWF=1 before enabling it again.

Bit 8 **ALRAE**: Alarm A enable

0: Alarm A disabled

1: Alarm A enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: PREDIV_S must be 0x00FF.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wakeup clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected
11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

40.6.9 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key
This byte is written by software.
Reading this byte always returns 0x00.
Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

40.6.10 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. If the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 × CALP) - CALM.

Refer to *Section 40.3.11: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to 1, the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 40.3.11: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 40.3.11: RTC smooth digital calibration.

Bits 12:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature must be used in conjunction with CALP. See *Section 40.3.11: RTC smooth digital calibration*.

40.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

40.6.12 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don’t care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don’t care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don’t care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don’t care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

40.6.13 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection*.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[14:1] are don't care in alarm A comparison. Only SS[0] is compared.
2: SS[14:2] are don't care in alarm A comparison. Only SS[1:0] are compared.
3: SS[14:3] are don't care in alarm A comparison. Only SS[2:0] are compared.
...
12: SS[14:12] are don't care in alarm A comparison. SS[11:0] are compared.
13: SS[14:13] are don't care in alarm A comparison. SS[12:0] are compared.
14: SS[14] is don't care in alarm A comparison. SS[13:0] are compared.
15: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Note: The overflow bit of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

40.6.14 RTC status register (RTC_SR)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUTF	Res.	ALRAF
													r		r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **WUTF**: Wakeup timer flag
This flag is set by hardware when the wakeup auto-reload counter reaches 0.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ALRAF**: Alarm A flag
This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared 2 APB clock cycles after setting their corresponding clear bit in the RTC_SCR register.

40.6.15 RTC masked interrupt status register (RTC_MISR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUTMF	Res.	ALRAMF
													r		r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **WUTMF**: Wakeup timer masked flag
This flag is set by hardware when the wakeup timer interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A interrupt occurs.

40.6.16 RTC status clear register (RTC_SCR)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUTF	Res.	CALRAF
													w		w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CWUTF**: Clear wakeup timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

41 Universal asynchronous receiver transmitter (UART)

This section describes the universal asynchronous receiver transmitter (UART).

41.1 UART introduction

The UART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The UART supports Half-duplex Single-wire communications, as well as LIN (local interconnection network), IrDA (infrared data association) SIR ENDEC specifications, and Modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

41.2 UART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte

- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection

41.3 UART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when UART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

41.4 UART implementation

The table below describes UART implementation on SR5E1x devices.

Table 462. UART features

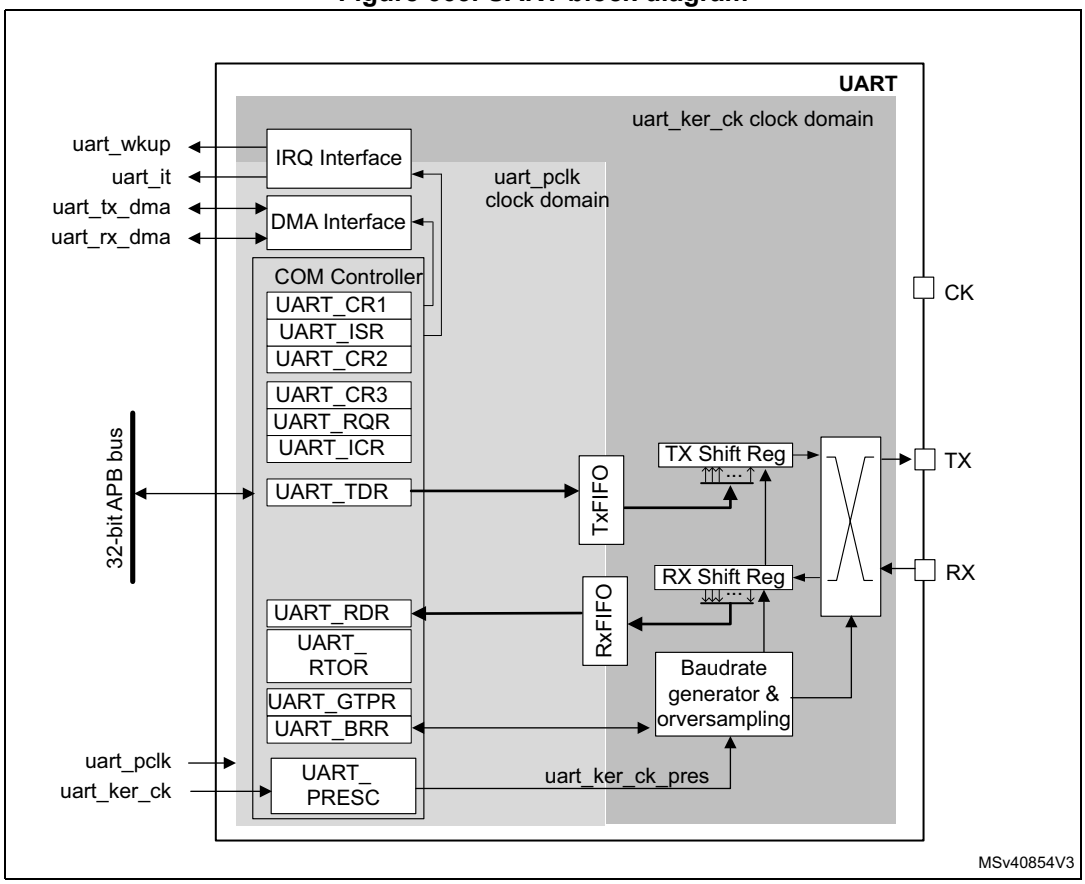
UART modes/features(1)	UART1/2/3
Hardware flow control for modem	-
Continuous communication using DMA	X
Multiprocessor communication	X
Synchronous mode (Master/Slave)	-
Smartcard mode	-
Single-wire Half-duplex communication	X
IrDA SIR ENDEC block	X
LIN mode	X
Dual clock domain and wakeup from low-power mode	X
Receiver timeout interrupt	X
Modbus communication	X
Auto baud rate detection	X
Driver Enable	-
UART data length	7, 8 and 9 bits
Tx/Rx FIFO	X
Tx/Rx FIFO size	8

1. X = supported.

41.5 UART functional description

41.5.1 UART block diagram

Figure 663. UART block diagram



The simplified block diagram given in *Figure 663* shows two fully-independent clock domains:

- The **uart_pclk** clock domain.

  The **uart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the UART registers are required.

- The **uart_ker_ck** kernel clock domain.

  The **uart_ker_ck** is the UART clock source. It is independent from **uart_pclk** and delivered by the RCC. The UART registers can consequently be written/read even when the **uart_ker_ck** clock is stopped.

  When the dual clock domain feature is disabled, the **uart_ker_ck** clock is the same as the **uart_pclk** clock.

There is no constraint between **uart_pclk** and **uart_ker_ck**: **uart_ker_ck** can be faster or slower than **uart_pclk**. The only limitation is the software ability to manage the communication fast enough.

41.5.2 UART signals

UART bidirectional communications

UART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)

  RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

- **TX** (Transmit Data Output)

  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In Single-wire mode, this I/O is used to transmit and receive data.

41.5.3 UART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the UART_CR1 register (see *Figure 664*):

- 7-bit character length: M[1:0] = ‘10’
- 8-bit character length: M[1:0] = ‘00’
- 9-bit character length: M[1:0] = ‘01’

Note: In 7-bit data length mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

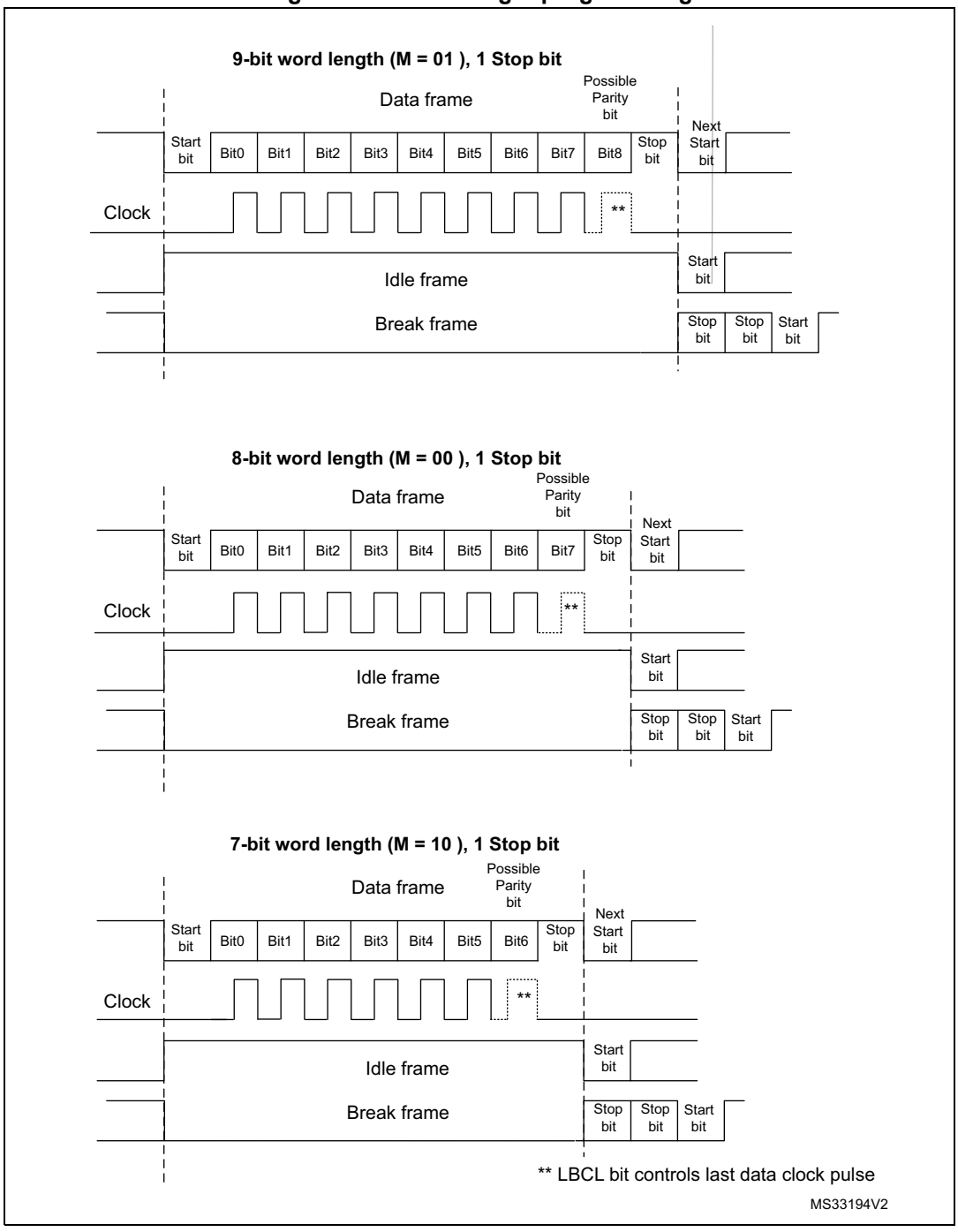
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 664. Word length programming



41.5.4 UART FIFOs and thresholds

The UART can operate in FIFO mode.

The UART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in UART_CR1 register (bit 29). This mode is supported only in UART mode.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the UART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bit fields in UART_CR3 control register.

In this case:

- The RXFT flag is set in the UART_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

  This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

  RXFTCFG data have been received: one data in UART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101’, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the UART_RDR). As a result, the next received data is not set the overrun flag.
- The TXFT flag is set in the UART_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

41.5.5 UART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an UART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the UART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (UART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the UART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

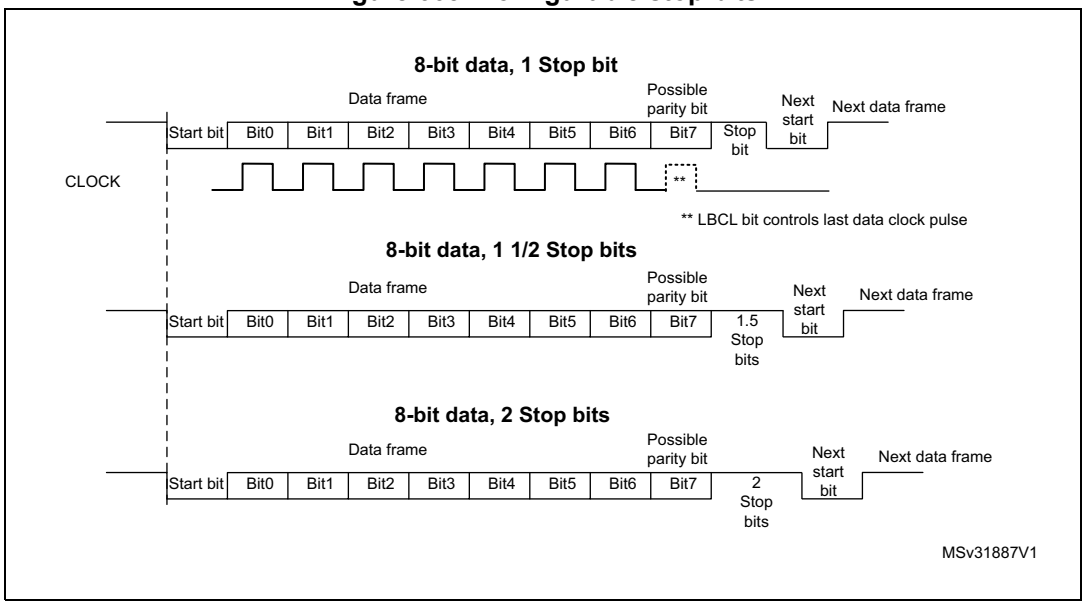
The number of stop bits to be transmitted with every character can be programmed in UART_CR2, bits 13,12.

- ***1 stop bit***: this is the default value of number of stop bits.
- ***2 stop bits***: this is supported by normal UART, Single-wire and Modem modes.

An idle frame transmission includes the stop bits.

A break transmission features 10 low bits (when M[1:0] = ‘00’) or 11 low bits (when M[1:0] = ‘01’) or 9 low bits (when M[1:0] = ‘10’) followed by 2 stop bits (see *Figure 665*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 665. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in UART_CR1 to define the word length.
2. Select the desired baud rate using the UART_BRR register.
3. Program the number of stop bits in UART_CR2.
4. Enable the UART by writing the UE bit in UART_CR1 register to 1.
5. Select DMA enable (DMAT) in UART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 41.5.10: UART multiprocessor communication*.
6. Set the TE bit in UART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the UART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data to the UART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data to the UART_TDR adds one data to the TXFIFO. Write operations to the UART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the UART_TDR register, wait until TC=1.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the UART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:

  – the data have been moved from the UART_TDR register to the shift register and the data transmission has started;

  – the UART_TDR register is empty;

  – the next data can be written to the UART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the UART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the UART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:

  – the TXFIFO is not full;

  – the UART_TDR register is empty;

  – the next data can be written to the UART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the UART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

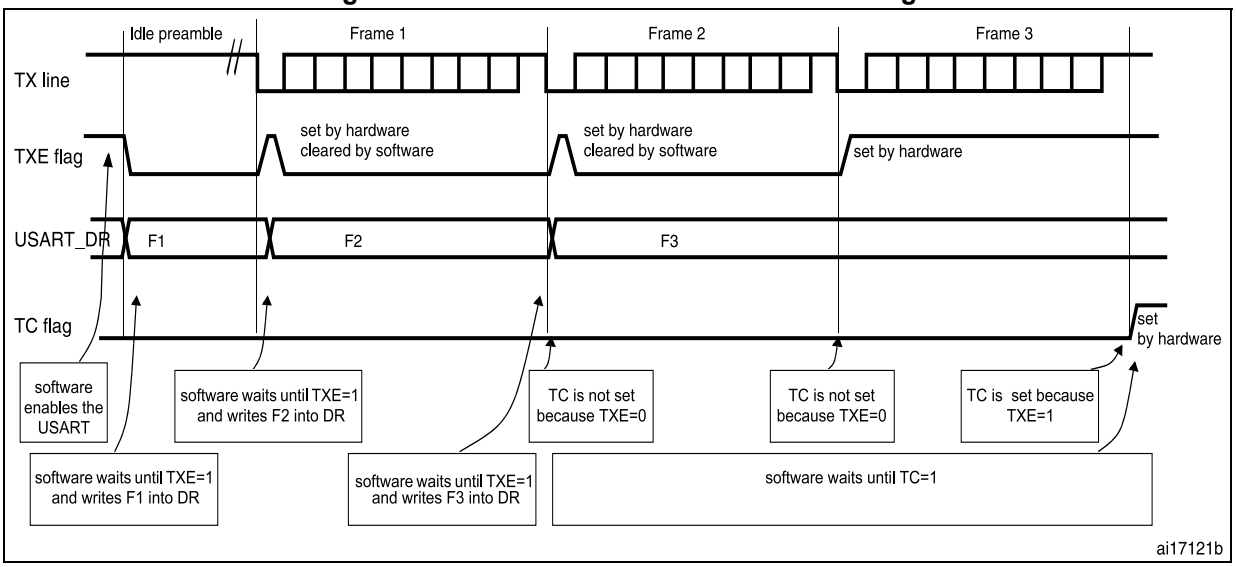
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write operation to UART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the UART_CR1 register.

After writing the last data to the UART_TDR register, it is mandatory to wait until TC is set before disabling the UART or causing the microcontroller to enter the low-power mode (see *Figure 666: TC/TXE behavior when transmitting*).

Figure 666. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 664*).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The UART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the UART to send an idle frame before the first data frame.

41.5.6 UART receiver

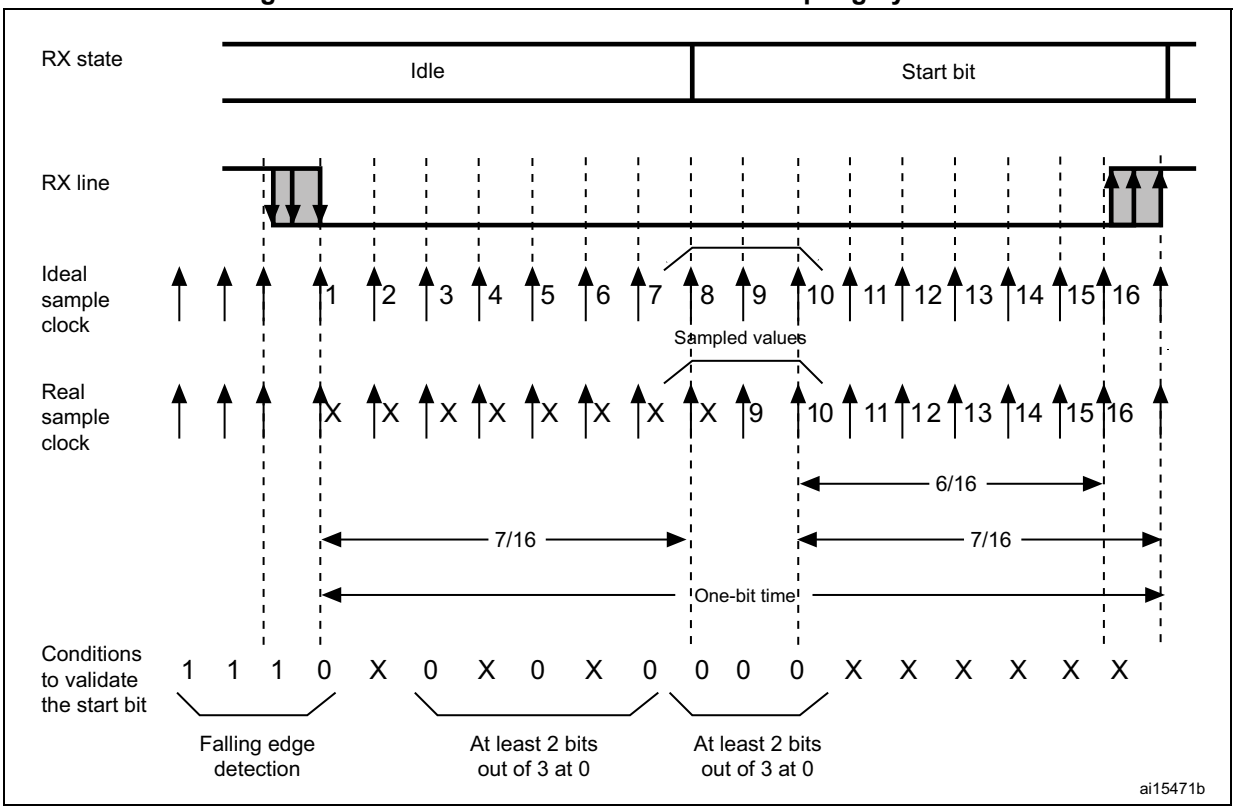
The UART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the UART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the UART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 667. Start bit detection when oversampling by 16 or 8



Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE=1, or RXFNE flag set and interrupt generated if RXFNEIE=1 if FIFO mode enabled) if the 3 sampled bits are at ‘0’ (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at ‘0’ and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at ‘0’).

The start bit is validated but the NE noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at ‘0’ (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at ‘0’.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an UART reception, data is shifted out by the least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in UART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register UART_BRR
3. Program the number of stop bits in UART_CR2.
4. Enable the UART by writing the UE bit in UART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in UART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 41.5.10: UART multiprocessor communication*.
6. Set the RE bit UART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the UART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty that is when there are data to be read from the RXFIFO.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the UART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to ‘1’ in the UART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from UART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to ‘1’ in UART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and

data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the UART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set.
  – the RDR content is not lost. The previous data is available by reading the UART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data cannot be transferred from the shift register to the UART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the UART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the UART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE=0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Chapter 11: Reset and clock control (RCC)*). The clock source must be selected through the UE bit before enabling the UART.

The clock source must be selected according to two criteria:

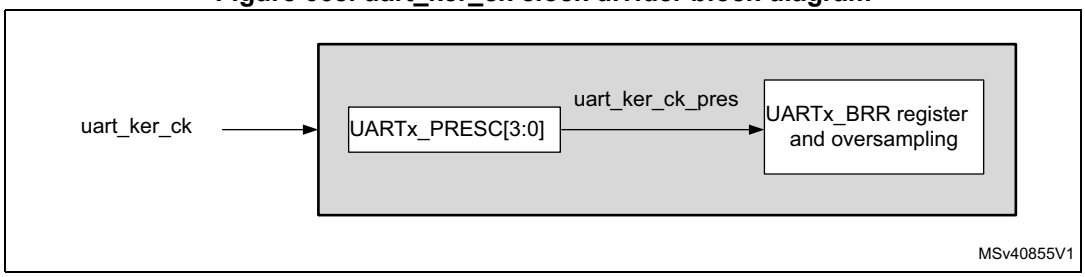
- Possible use of the UART in low-power mode.
- Communication speed.

The clock source frequency is uart_ker_ck.

When the dual clock domain and the wakeup from low-power mode features are supported, the uart_ker_ck clock source can be configurable in the RCC (see *Chapter 11: Reset and clock control (RCC)*). Otherwise the uart_ker_ck clock is the same as uart_pclk.

The uart_ker_ck clock can be divided by a programmable factor, defined in the UART_PRESC register.

Figure 668. uart_ker_ck clock divider block diagram



Some uart_ker_ck sources enable the UART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selected, the UART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the UART_RDR register or by DMA.

For the other clock sources, the system must be active to enable UART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the UART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 669* and *Figure 670*).

Depending on your application:

- select oversampling by 8 (OVER8=1) to achieve higher speed (up to uart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 41.5.8: Tolerance of the UART receiver to clock deviation on page 1526*)
- select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum

uart_ker_ck_pres/16 (where uart_ker_ck_pres is the UART input clock divided by a prescaler).

Programming the ONEBIT bit in the UART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on your application:

  – select the three sample majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Figure 463*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 41.5.8: Tolerance of the UART receiver to clock deviation on page 1526*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the UART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the UART_CR3 register.

The NE bit is reset by setting NECF bit in UART_ICR register.

Note: Noise error is not supported in SPI mode.

Oversampling by 8 is not available in the IrDA and LIN modes. In those modes, the OVER8 bit is forced to '0' by hardware.

Figure 669. Data sampling when oversampling by 16

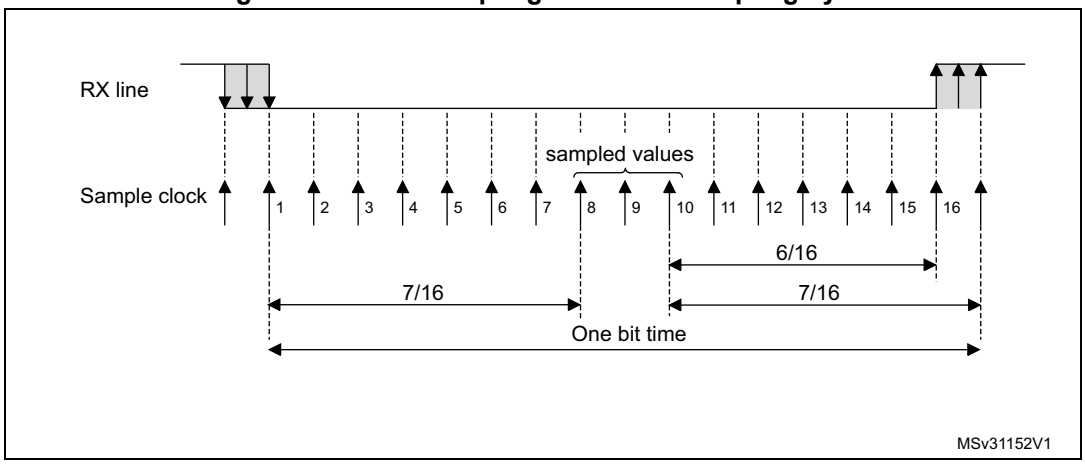


Figure 670. Data sampling when oversampling by 8

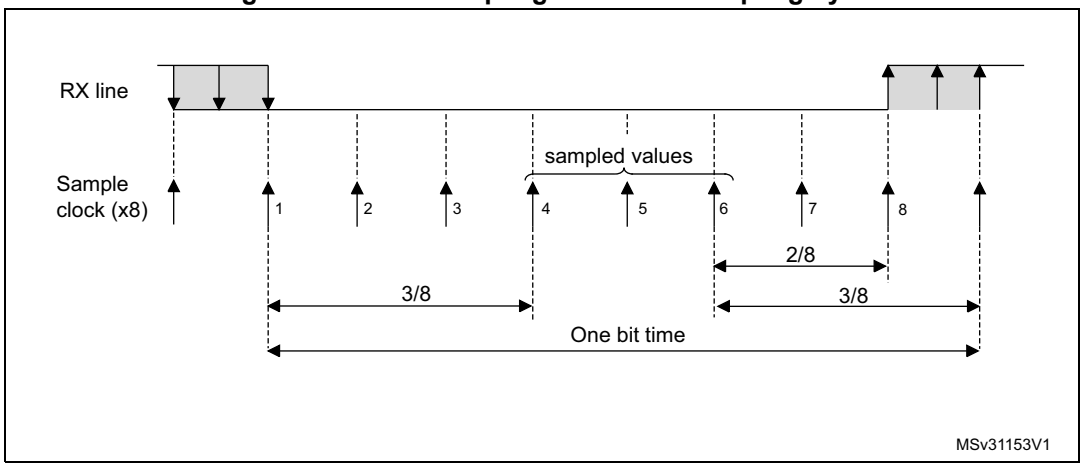


Table 463. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware.
- the invalid data is transferred from the Shift register to the UART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the UART_CR3 register.

The FE bit is reset by writing ‘1’ to the FECF in the UART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of UART_CR: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

41.5.7 UART baud rate generation

The baud rates for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the UART_BRR register.

Equation 20: baud rate for standard UART (SPI mode included) (OVER8 = '0' or '1')

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{uart_ker_ckpres}}{\text{UARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{uart_ker_ckpres}}{\text{UARTDIV}}$$

Equation 21: baud rate in LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{uart_ker_ckpres}}{\text{UARTDIV}}$$

UARTDIV is an unsigned fixed point number that is coded on the UART_BRR register.

- When OVER8 = 0, BRR = UARTDIV.
- When OVER8 = 1
  - BRR[2:0] = UARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = UARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to UART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, UARTDIV must be greater than or equal to 16.

How to derive UARTDIV from UART_BRR register values

Example 1

To obtain 9600 baud with uart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  UARTDIV = 8 000 000/9600
  BRR = UARTDIV = 833d = 0341h
- In case of oversampling by 8:
  UARTDIV = 2 * 8 000 000/9600
  UARTDIV = 1666,66 (1667d = 683h)
  BRR[3:0] = 3h >>1 = 1h
  BRR = 0x681

Example 2

To obtain 921.6 Kbaud with uart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  UARTDIV = 48 000 000/921 600
  BRR = UARTDIV = 52d = 34h
- In case of oversampling by 8:
  UARTDIV = 2 * 48 000 000/921 600
  UARTDIV = 104 (104d = 68h)
  BRR[3:0] = UARTDIV[3:0] >> 1 = 8h >> 1 = 4h
  BRR = 0x64

41.5.8 Tolerance of the UART receiver to clock deviation

The UART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the UART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator

- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{UART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUART}}{9 \times Tbit}$$

t_{WUUART} is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The UART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 464*, *Table 465*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the UART_CR1 register.
- Oversampling by 8 or 16 defined by the OVER8 bit in the UART_CR1 register.
- Bits BRR[3:0] of UART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the UART_CR3 register.

Table 464. Tolerance of the UART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 465. Tolerance of the UART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 464 and Table 465 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

41.5.9 UART Auto baud rate detection

The UART can detect and automatically set the UART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from uart_ker_ck_pres/65535 and uart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from uart_ker_ck_pres/65535 and uart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the UART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at ‘1’.

  In this case the UART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the UART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the UART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the UART_CR2 register. The UART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the UART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a ‘0’).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection has to be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in UART_ISR register.

Note: The BRR value might be corrupted if the UART is disabled (UE=0) during an auto baud rate operation.

41.5.10 UART multiprocessor communication

It is possible to perform UART multiprocessor communications (with several UARTs connected in a network). For instance one of the UARTs can be the master with its TX output connected to the RX inputs of the other UARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant UART service overhead for all non addressed receivers.

The non-addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the UART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two uart_ker_ck cycles), otherwise Mute mode might remain active.

When the Mute mode is enabled:

- none of the reception status bits can be set.
- all the receive interrupts are inhibited.
- the RWU bit in UART_ISR register is set to ‘1’. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the UART_RQR register, under certain conditions.

The UART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the UART_CR1 register:

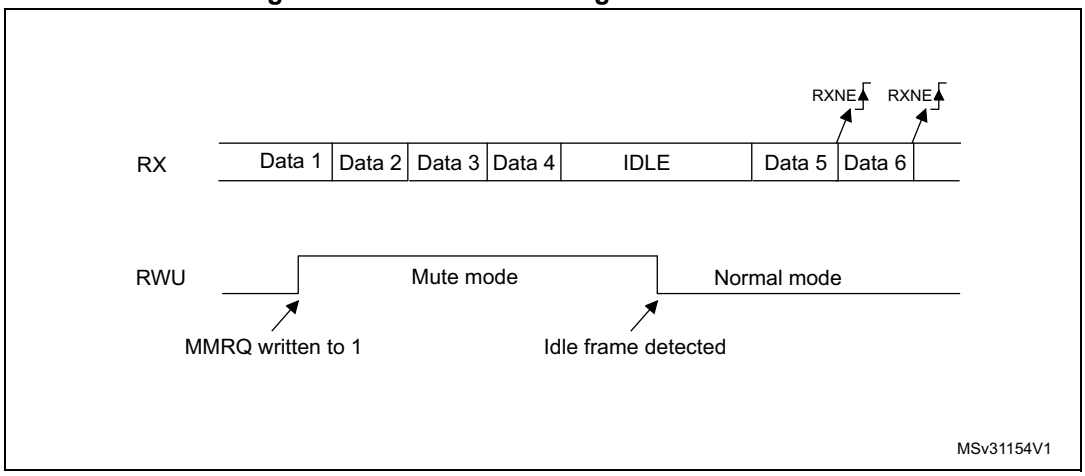
- Idle Line detection if the WAKE bit is reset.
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The UART enters Mute mode when the MMRQ bit is written to ‘1’ and the RWU is automatically set.

The UART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the UART_ISR register. An example of Mute mode behavior using Idle line detection is given in *Figure 671*.

Figure 671. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, Mute mode is not entered (RWU is not set).

If the UART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the UART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The UART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the UART enters Mute mode. When FIFO management is enabled, the software has to ensure that there is at least one empty location in the RXFIFO before entering Mute mode.

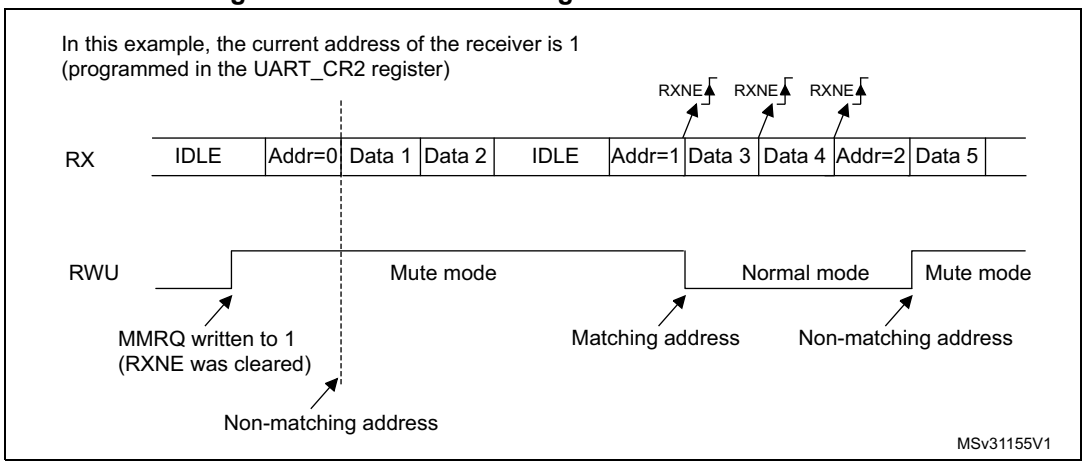
The UART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The UART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in Mute mode

An example of Mute mode behavior using address mark detection is given in *Figure 672*.

Figure 672. Mute mode using address mark detection



41.5.11 UART Modbus communication

The UART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a Half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The UART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the UART_CR2 register and the RTOIE in the UART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The UART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE=1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

41.5.12 UART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the UART_CR1 register. Depending on the frame length defined by the M bits, the possible UART frame formats are as listed in *Table 466*.

Table 466. UART frame formats

M bits	PCE bit	UART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits are set, the parity bit is equal to 0 if even parity is selected (PS bit in UART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in UART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the UART_ISR register and an interrupt is generated if PEIE is set in the UART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the UART_ICR register.

Parity generation in transmission

If the PCE bit is set in UART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

41.5.13 UART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to the *Section 41.4: UART implementation on page 1510*.

The LIN mode is selected by setting the LINEN bit in the UART_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the UART_CR2 register.
- STOP[1:0], SCEN, HDSEL and IREN in the UART_CR3 register.

LIN transmission

The procedure described in the *Section 41.5.4* has to be applied for LIN Master transmission. It must be the same as for normal UART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0 bits as a break character. Then two bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal UART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in UART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in UART_CR2) or 11 (when LBDL=1 in UART_CR2) consecutive bits are detected as ‘0, and are followed by a delimiter character, the LBDF flag is set in UART_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1 is sampled before the 10 or 11 has occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal UART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (that is stop bit detected at ‘0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the *Figure 673: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 1535*.

Examples of break frames are given on *Figure 674: Break detection in LIN mode vs. Framing error detection on page 1536*.

Figure 673. Break detection in LIN mode (11-bit break length - LBDL bit is set)

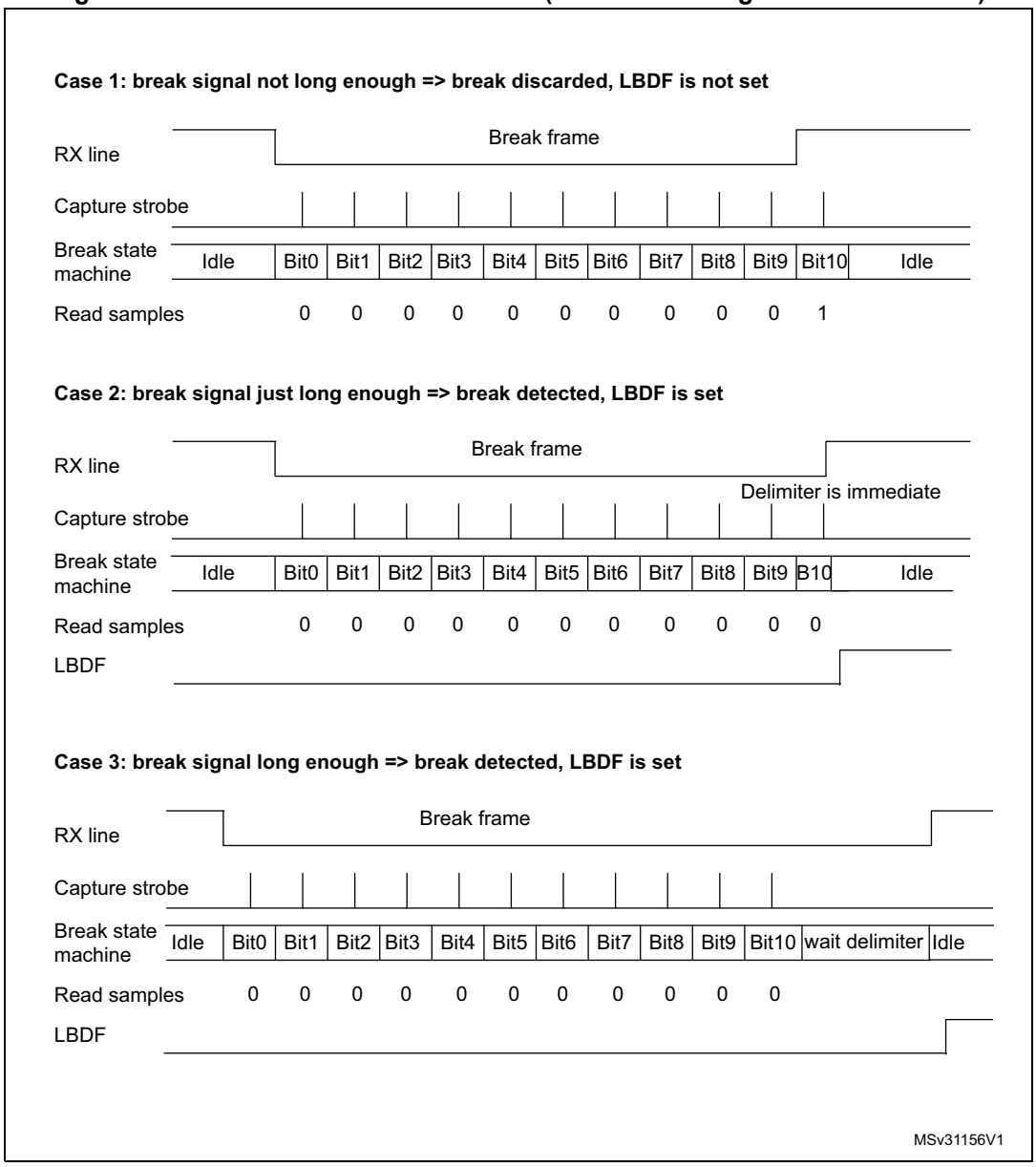
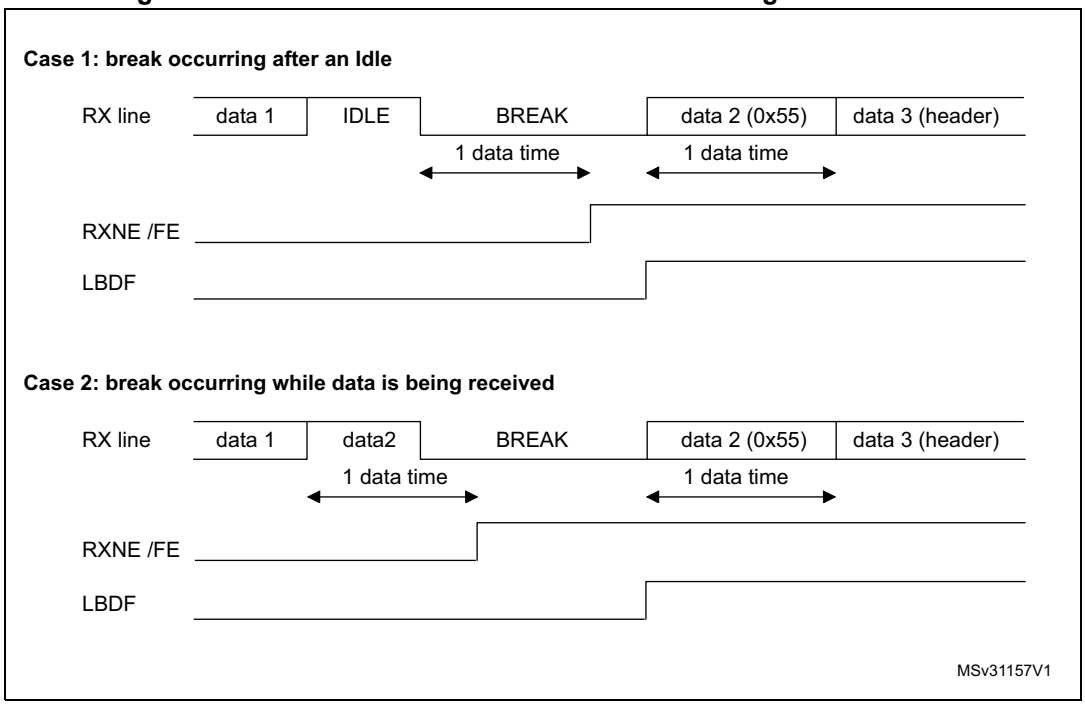


Figure 674. Break detection in LIN mode vs. Framing error detection



41.5.14 UART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the UART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the UART_CR2 register.
- SCEN and IREN bits in the UART_CR3 register.

The UART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in UART_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal UART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

41.5.15 UART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the UART_CR2 control register.

The timeout duration is programmed using the RTO bit fields in the UART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = ‘00’ or STOP = ‘11’
- from the end of the second stop bit if STOP = ‘10’.
- from the beginning of the stop bit if STOP = ‘01’.

When the timeout duration has elapsed, the RTOF flag in the UART_ISR register is set. A timeout is generated if RTOIE bit in UART_CR1 register is set.

41.5.16 UART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to the *Section 41.4: UART implementation*.

IrDA mode is selected by setting the IREN bit in the UART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the UART_CR2 register,
- SCEN and HDSEL bits in the UART_CR3 register.

The IrDA SIR physical layer specifies the use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 675*).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from UART. The output pulse stream is transmitted to an external output driver and infrared LED. UART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the UART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the UART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the UART is receiving decoded data from the UART), data on the TX from the UART to IrDA is not encoded. While receiving data, transmission has to be avoided as the data to be transmitted can be corrupted.
- A ‘0’ is transmitted as a high pulse and a ‘1’ is transmitted as a ‘0’. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 676*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for UART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the UART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC=0.

- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the UART_CR2 register must be configured to ‘1 stop bit’.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection, the UART has to discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the UART_GTPR).

Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time has to be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).

Figure 675. IrDA SIR ENDEC block diagram

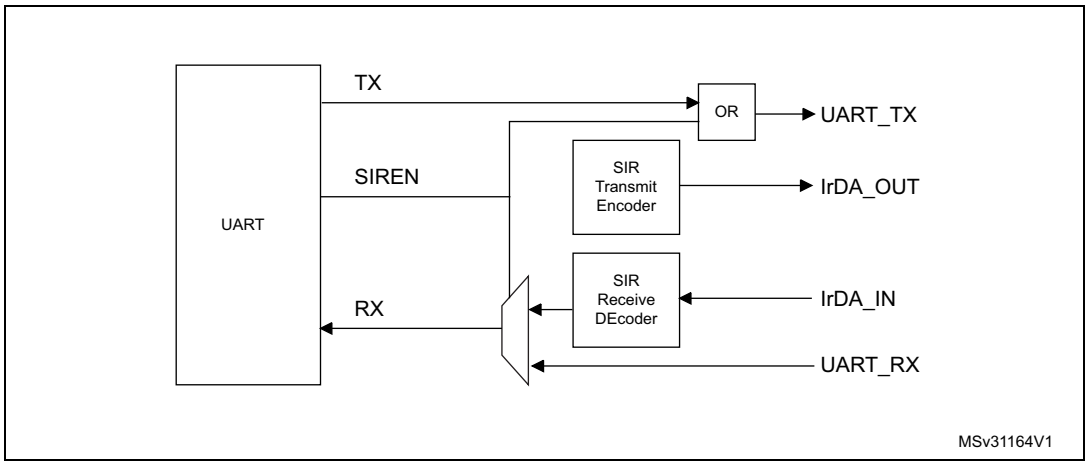
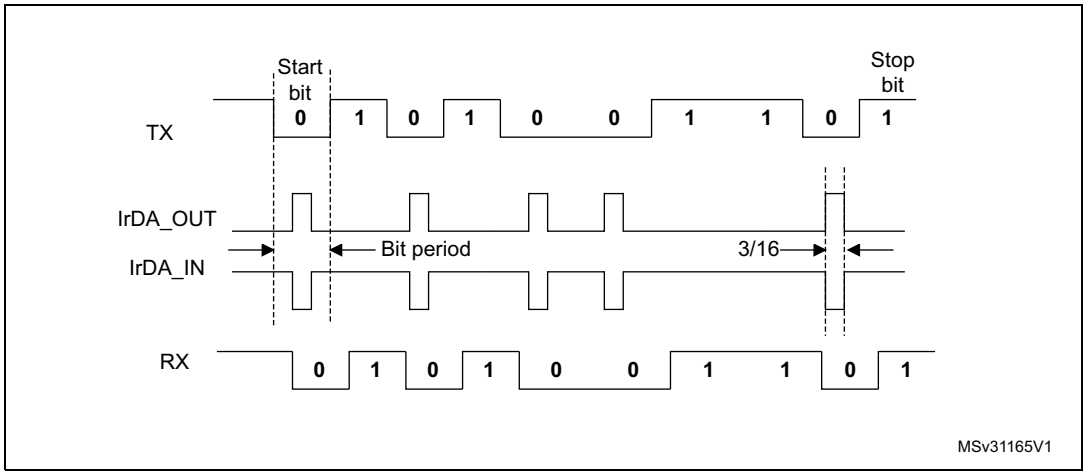


Figure 676. IrDA data modulation (3/16) - Normal mode



41.5.17 Continuous communication using UART and DMA

The UART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to the Section 41.4: UART implementation on page 1510 to determine if the DMA mode is supported. If DMA is not supported, use the UART as explained in the Section 41.5.6. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the UART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the UART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *Chapter 23: Direct memory access controller (DMA)*) to the UART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for UART transmission, use the following procedure (x denotes the channel number):

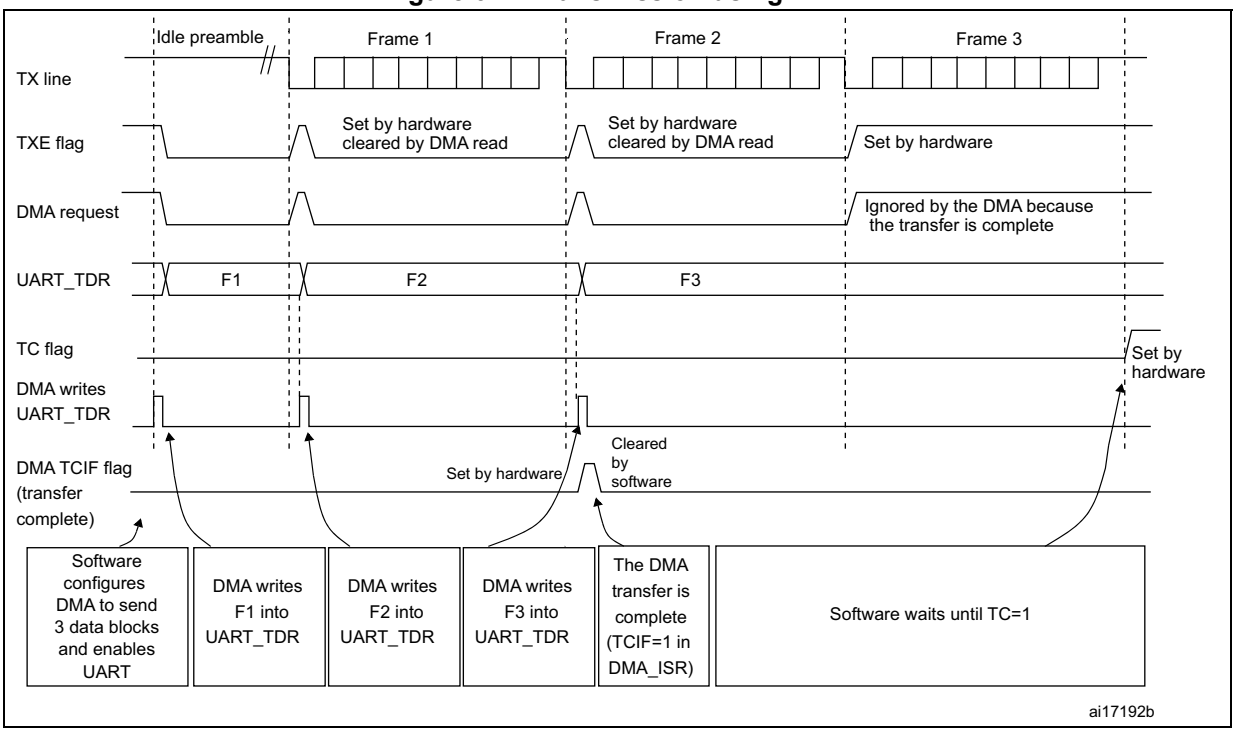
1. Write the UART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the UART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register.
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the UART_ISR register by setting the TCCF bit in the UART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the UART

communication is complete. This is required to avoid corrupting the last transmission before disabling the UART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 677. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (that is TXFNF = 1).

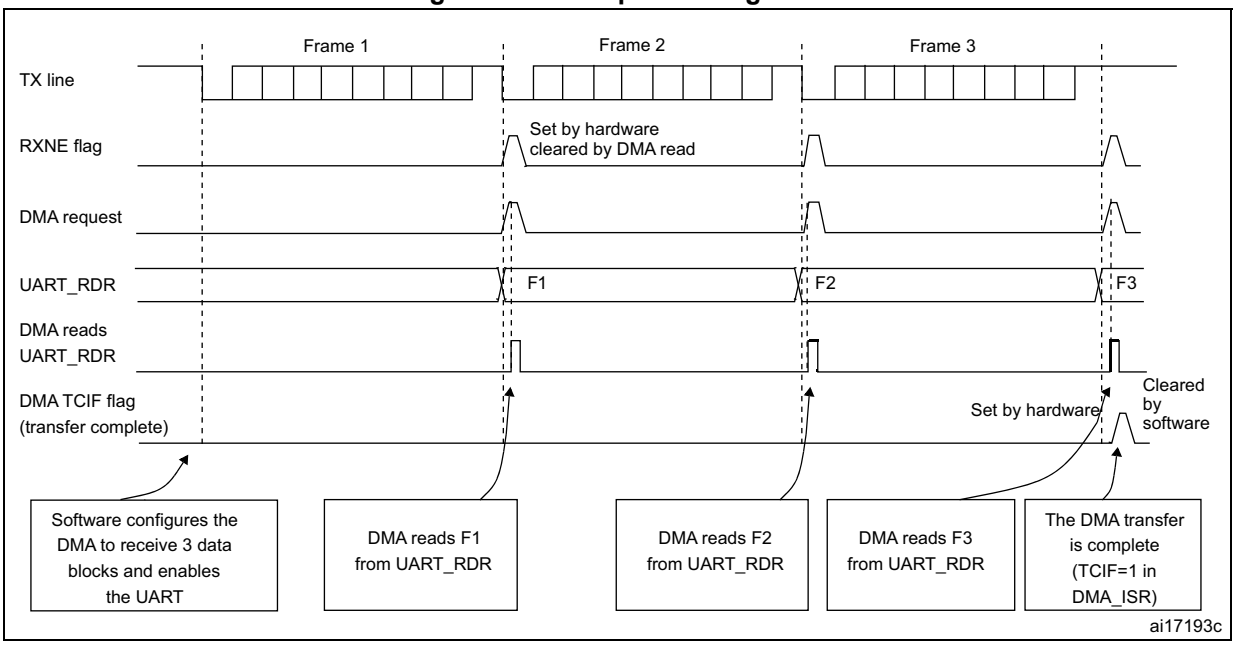
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in UART_CR3 register. Data are loaded from the UART_RDR register to an SRAM area configured using the DMA peripheral (refer to *Chapter 23: Direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for UART reception, use the following procedure:

1. Write the UART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from UART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 678. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (that is RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the UART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

41.5.18 UART low-power management

The UART has advanced low-power mode functions, that enables transferring properly data even when the uart_pclk clock is disabled.

The UART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the uart_pclk is gated, the UART provides a wakeup interrupt (**uart_wkup**) if a specific action requiring the activation of the **uart_pclk** clock is needed:

- If FIFO mode is disabled

  uart_pclk clock has to be activated to empty the UART data register.

  In this case, the uart_wkup interrupt source is RXNE set to '1'. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  uart_pclk clock has to be activated to:
  - to fill the TXFIFO
  - or to empty the RXFIFO

  In this case, the uart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the uart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode enables doing as many UART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **uart_wkup** interrupt can be selected through the WUS bit fields.

When the wakeup event is detected, the WUF flag is set by hardware and a **uart_wkup** interrupt is generated if the WUFIE bit is set. In this case the **uart_wkup** interrupt is not mandatory and setting the WUF being is sufficient to wake up the MCU from low-power mode.

Note: Before entering low-power mode, make sure that no UART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the UART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when Mute mode is enabled.

Using Mute mode with low-power mode

If the UART is put into Mute mode before entering low-power mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wakes up from low-power mode.

Note: When FIFO management is enabled, Mute mode can be used with wakeup from low-power mode without any constraints (that is the two points mentioned above about Mute and low-power mode are valid only when FIFO management is disabled).

Wakeup from low-power mode when UART kernel clock (uart_ker_ck) is OFF in low-power mode

If during low-power mode, the uart_ker_ck clock is switched OFF when a falling edge on the UART receive line is detected, the UART interface requests the uart_ker_ck clock to be switched ON thanks to the uart_ker_ck_req signal. uart_ker_ck is then used for the frame reception.

If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, uart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wakeup event programmed to “address match detection” and FIFO management disabled.

Figure 679 shows the UART behavior when the wakeup event is verified.

Figure 679. Wakeup event verified (wakeup event = address match, FIFO disabled)

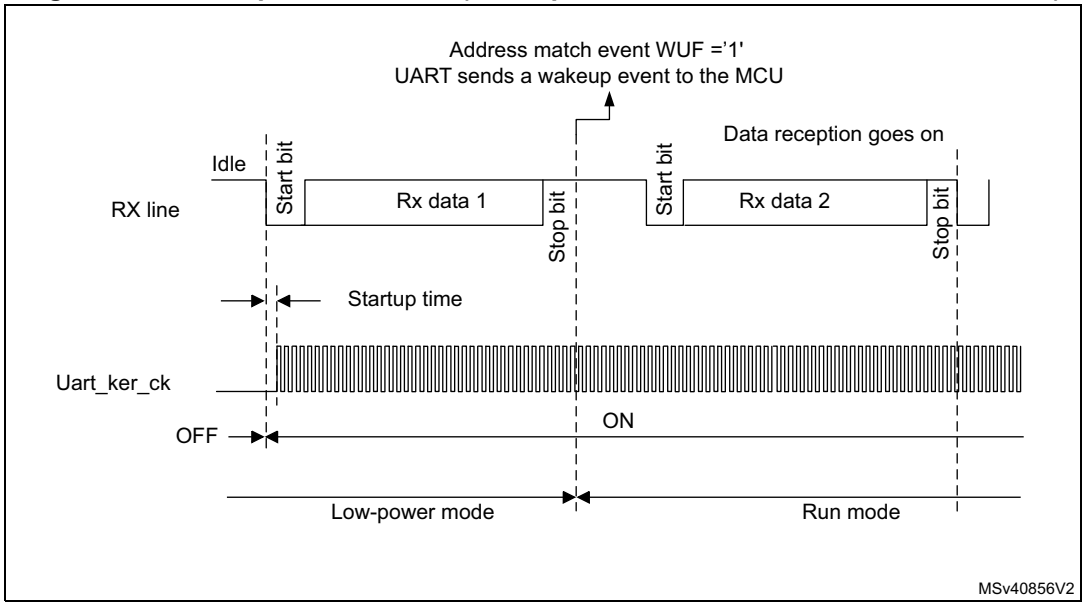
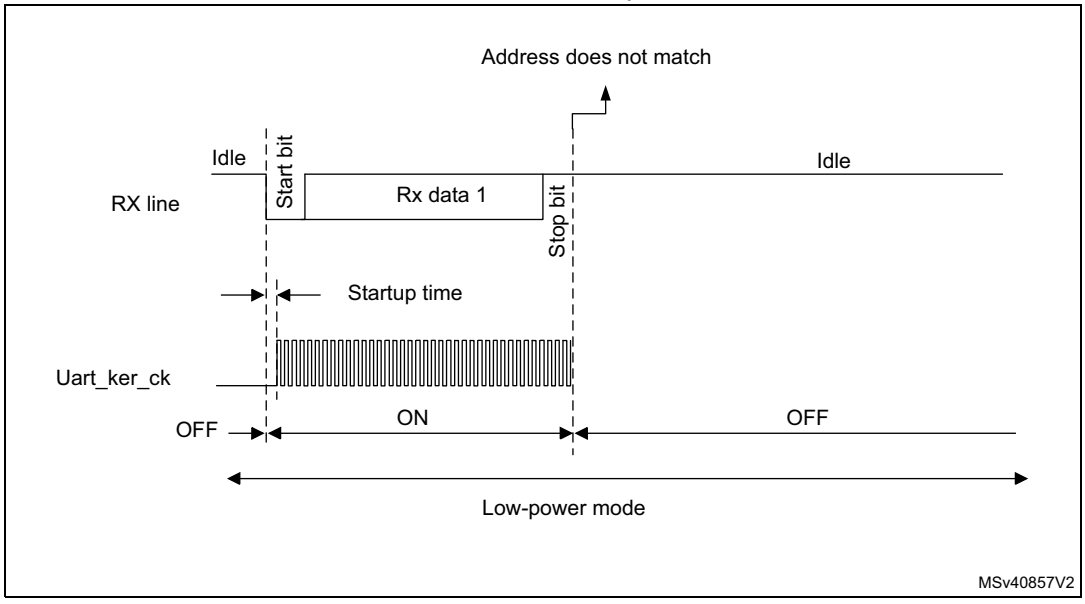


Figure 680 shows the UART behavior when the wakeup event is not verified.

Figure 680. Wakeup event not verified (wakeup event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as wakeup event. If the wakeup event is the start bit detection, the UART sends the wakeup event to the MCU at the end of the start bit.

Determining the maximum UART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the UART receiver tolerance (see *Section 41.5.8: Tolerance of the UART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 464: Tolerance of the UART receiver when BRR [3:0] = 0000*, the UART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < UART receiver tolerance

$D_{WUmax} = t_{WUUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = = t_{WUUART} / (11 \times D_{WUmax})$

where t_{WUUART} is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the uart_ker_ck inaccuracy.

For example, if IRCOSC is used as uart_ker_ck, and the IRCOSC inaccuracy is of 1%, then we obtain:

t_{WUUART} = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% - 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wakeup correctly from low-power mode is: 1/11.32 µs = 88.36 Kbaud.

41.6 UART interrupts

During UART communications, an interrupt (uart_it) can be generated by different events. The UART block can also generate a wakeup interrupt (uart_wkup).

Refer to *Table 467* for a detailed description of all UART interrupt requests.

Table 467. UART interrupt requests

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				uart_it	uart_wkup
Transmit data register empty	TXE	TXEIE	TXE cleared when a data is written in TDR	YES	NO
Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFNF cleared when TXFIFO is full.	YES	NO
Transmit FIFO Empty	TXFE	TXFEIE	TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.	YES	YES

Table 467. UART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				uart_it	uart_wkup
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES
CTS interrupt	CTSIF	CTSIE	CTSIF cleared by software by setting CTSCF bit.	YES	NO
Transmission Complete	TC	TCIE	TC cleared when a data is written in TDR or by setting TCCF bit.	YES	NO
Transmission Complete Before Guard Time	TCBGT	TCBGTIE	TCBGT cleared when a data is written in TDR or by setting TCBGTCF bit.	YES	NO
Receive data register not empty (data ready to be read)	RXNE	RXNEIE	RXNE cleared by reading RDR or by setting RXFRQ bit.	YES	YES
Receive FIFO Not Empty	RXFNE	RXFNEIE	RXFNE cleared when the RXFIFO is empty or by setting RXFRQ bit.	YES	YES
Receive FIFO Full	RXFF(1)	RXFFIE	RXFF cleared when the RXFIFO contains at least one data.	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold	YES	YES
Overrun error detected	ORE	RX-NEIE/RX FNEIE	ORE cleared by setting ORECF bit.	YES	NO
Idle line detected	IDLE	IDLEIE	IDLE cleared by setting IDLECF bit.	YES	NO
Parity error	PE	PEIE	PE cleared by setting PECF bit.	YES	NO
LIN break	LBDF	LBDIE	LBDF cleared by setting LBDCF bit.	YES	NO
Noise error, Overrun error and Framing Error in multibuffer communication.	NE or ORE or FE	EIE	NE cleared by setting NECF bit. ORE cleared by setting ORECF bit. FE flag cleared by setting FECF bit.	YES	NO
Character match	CMF	CMIE	CMF cleared by setting CMCF bit.	YES	NO
Receiver timeout	RTOF	RTOFIE	RTOF cleared by setting RTOCCF bit.	YES	NO

Table 467. UART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				uart_it	uart_wkup
End of Block	EOBF	EOBIE	EOBF is cleared by setting EOBCF bit.	YES	NO
Wakeup from low-power mode	WUF	WUFIE	WUF is cleared by setting WUCF bit.	YES	YES
SPI slave underrun error	UDR	EIE	UDR is cleared by setting UDRCF bit.	YES	NO
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold.	YES	YES

1. RXFF flag is asserted if the UART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in UART_RDR.

41.7 UART registers

Refer to *Section 1.2: Register conventions* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

41.7.1 UART memory map

Table 468. UART register memory map

Offset	Register name
0x00	*UART control register 1 [alternate] (UART_CR1)*
0x00	*UART control register 1 [alternate] (UART_CR1)*
0x04	*UART control register 2 (UART_CR2)*
0x08	*UART control register 3 (UART_CR3)*
0x0C	*UART baud rate register (UART_BRR)*
0x10	*UART guard time and prescaler register (UART_GTPR)*
0x14	*UART receiver timeout register (UART_RTOR)*
0x18	*UART request register (UART_RQR)*
0x1C	*UART interrupt and status register [alternate] (UART_ISR)*
0x1C	*UART interrupt and status register [alternate] (UART_ISR)*
0x20	*UART interrupt flag clear register (UART_ICR)*
0x24	*UART receive data register (UART_RDR)*

Table 468. UART register memory map (continued)

Offset	Register name
0x28	*UART transmit data register (UART_TDR)*
0x2C	*UART prescaler register (UART_PRESC)*

41.7.2 UART control register 1 [alternate] (UART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated when RXFF=1 in the UART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated when TXFE=1 in the UART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bit field can only be written when the UART is disabled (UE=0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = ‘00’: 1 start bit, 8 Data bits, n Stop bit
M[1:0] = ‘01’: 1 start bit, 9 Data bits, n Stop bit
M[1:0] = ‘10’: 1 start bit, 7 Data bits, n Stop bit
This bit can only be written when the UART is disabled (UE=0).

Note: In 7-bit data length mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of Block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated when the EOBF flag is set in the UART_ISR register

Note: If the UART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated when the RTOF bit is set in the UART_ISR register.

Note: If the UART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 41.4: UART implementation on page 1510.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bit field can only be written when the UART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the deactivation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the UART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bit field can only be written when the UART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the UART is disabled (UE=0).

Note: In LIN and IrDA modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated when the CMF bit is set in the UART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the UART Mute mode function. When set, the UART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the UART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the UART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bit field can only be written when the UART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bit field can only be written when the UART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bit field can only be written when the UART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated whenever PE=1 in the UART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated whenever TXFNF =1 in the UART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated whenever TC=1 in the UART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated whenever ORE=1 or RXFNE=1 in the UART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated whenever IDLE=1 in the UART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit (‘0’ followed by ‘1’) sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to ‘1’. To ensure the required duration, the software can poll the TEACK bit in the UART_ISR register.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: UART enable in low-power mode

When this bit is cleared, the UART cannot wake up the MCU from low-power mode.

When this bit is set, the UART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: UART not able to wake up the MCU from low-power mode.

1: UART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.

If the UART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 0 **UE**: UART enable

When this bit is cleared, the UART prescalers and outputs are stopped immediately, and all current operations are discarded. The UART configuration is kept, but all the UART_ISR status flags are reset. This bit is set and cleared by software.

0: UART prescaler and outputs disabled, low-power mode

1: UART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the UART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

41.7.3 UART control register 1 [alternate] (UART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bit field can only be written when the UART is disabled (UE=0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = ‘00’: 1 start bit, 8 Data bits, n Stop bit

M[1:0] = ‘01’: 1 start bit, 9 Data bits, n Stop bit

M[1:0] = ‘10’: 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the UART is disabled (UE=0).

Note: In 7-bit data length mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated when the EOBF flag is set in the UART_ISR register

Note: If the UART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: UART interrupt generated when the RTOF bit is set in the UART_ISR register.

Note: If the UART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 41.4: UART implementation on page 1510.

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (driver enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bit field can only be written when the UART is disabled (UE=0).

Note: If the driver enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (driver enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the UART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bit field can only be written when the UART is disabled (UE=0).

Note: If the driver enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 15 **OVER8**: Oversampling mode
0: Oversampling by 16
1: Oversampling by 8
This bit can only be written when the UART is disabled (UE=0).
Note: In LIN, IrDA modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated when the CMF bit is set in the UART_ISR register.

Bit 13 **MME**: Mute mode enable
This bit enables the UART mute mode function. When set, the UART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.
0: Receiver in active mode permanently
1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length
This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).
This bit can only be written when the UART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method
This bit determines the UART wakeup method from mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bit field can only be written when the UART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bit field can only be written when the UART is disabled (UE=0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bit field can only be written when the UART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated whenever PE=1 in the UART_ISR register

Bit 7 **TXEIE**: Transmit data register empty
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated whenever TXE =1 in the UART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated whenever TC=1 in the UART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated whenever ORE=1 or RXNE=1 in the UART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated whenever IDLE=1 in the UART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled
Note: During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the UART_ISR register.

Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: UART enable in low-power mode
When this bit is cleared, the UART cannot wake up the MCU from low-power mode.
When this bit is set, the UART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: UART not able to wake up the MCU from low-power mode.
1: UART able to wake up the MCU from low-power mode.
Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.
If the UART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 0 **UE**: UART enable
When this bit is cleared, the UART prescalers and outputs are stopped immediately, and all current operations are discarded. The UART configuration is kept, but all the UART_ISR status flags are reset. This bit is set and cleared by software.
0: UART prescaler and outputs disabled, low-power mode
1: UART enabled
Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the UART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

41.7.4 UART control register 2 (UART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the UART node

ADD[7:4]:

These bits give the address of the UART node or a character code to be recognized.

They are used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or low-power mode. The MSB of the character sent by the transmitter must be equal to 1. They can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.

These bits can only be written when reception is disabled (RE = 0) or the UART is disabled (UE=0).

ADD[3:0]:

These bits give the address of the UART node or a character code to be recognized.

They are used for wakeup with address mark detection, in multiprocessor communication during Mute mode or low-power mode.

These bits can only be written when reception is disabled (RE = 0) or the UART is disabled (UE=0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the UART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the UART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bit field can only be written when ABREN = 0 or the UART is disabled (UE=0).

Note: If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the UART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 20 **ABREN**: Auto baud rate enable
This bit is set and cleared by software.
0: Auto baud rate detection is disabled.
1: Auto baud rate detection is enabled.
Note: If the UART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 19 **MSBFIRST**: Most significant bit first
This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bit field can only be written when the UART is disabled (UE=0).

Bit 18 **DATAINV:** Binary data inversion
This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bit field can only be written when the UART is disabled (UE=0).

Bit 17 **TXINV:** TX pin active level inversion
This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: TX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).
This enables the use of an external inverter on the TX line.
This bit field can only be written when the UART is disabled (UE=0).

Bit 16 **RXINV:** RX pin active level inversion
This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: RX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).
This enables the use of an external inverter on the RX line.
This bit field can only be written when the UART is disabled (UE=0).

Bit 15 **SWAP:** Swap TX/RX pins
This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bit field can only be written when the UART is disabled (UE=0).

Bit 14 **LINEN**: LIN mode enable
This bit is set and cleared by software.
0: LIN mode disabled
1: LIN mode enabled
The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the UART_RQR register, and to detect LIN Sync breaks.
This bit field can only be written when the UART is disabled (UE=0).
Note: If the UART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 13:12 **STOP[1:0]**: stop bits

These bits are used for programming the stop bits.
00: 1 stop bit
01: 0.5 stop bit.
10: 2 stop bits
11: 1.5 stop bits
This bit field can only be written when the UART is disabled (UE=0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the SCLK pin.
0: SCLK pin disabled
1: SCLK pin enabled
This bit can only be written when the UART is disabled (UE=0).

Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship
0: Steady low value on SCLK pin outside transmission window
1: Steady high value on SCLK pin outside transmission window
This bit can only be written when the UART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 669* and *Figure 670*)
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge
This bit can only be written when the UART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.
0: The clock pulse of the last data bit is not output to the SCLK pin
1: The clock pulse of the last data bit is output to the SCLK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the UART_CR1 register.

This bit can only be written when the UART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).
0: Interrupt is inhibited
1: An interrupt is generated whenever LBDF=1 in the UART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 5 **LBDL**: LIN break detection length
This bit is for selection between 11 bit or 10 bit break detection.
0: 10-bit break detection
1: 11-bit break detection
This bit can only be written when the UART is disabled (UE=0).
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 4 **ADDM7**: 7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the UART is disabled (UE=0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:
When the DIS_NSS bit is set, the NSS pin input is ignored.
0: SPI slave selection depends on NSS input pin.
1: SPI slave is always selected and NSS input pin is ignored.
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous Slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Note: The CPOL, CPHA and LBCL bits must not be written while the transmitter is enabled.

41.7.5 UART control register 3 (UART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS[1:0]		SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Remaining combinations: Reserved

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Remaining combinations: Reserved

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated whenever TCBGT=1 in the UART_ISR register
Note: If the UART does not support the Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: UART interrupt generated whenever WUF=1 in the UART_ISR register
Note: WUFIE must be set before entering in low-power mode.

If the UART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 21:20 **WUS[1:0]**: Wakeup from low-power mode interrupt flag selection

This bit field specifies the event which activates the WUF (Wakeup from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bit field can only be written when the UART is disabled (UE=0).

If the UART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bit field specifies the number of retries for transmission and reception in Smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bit field must be programmed only when the UART is disabled (UE=0).

When the UART is enabled (UE=1), this bit field may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmit mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the UART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the UART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 41.4: UART implementation on page 1510.

Bit 13 **DDRE**: DMA Disable on Reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the UART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the UART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data is written directly in UART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the UART is disabled (UE=0).

Note: This control bit enables checking the communication flow w/o reading the data

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the UART is disabled (UE=0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF=1 in the UART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

This bit can only be written when the UART is disabled (UE=0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the UART is disabled (UE=0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

This bit field can only be written when the UART is disabled (UE=0).

Note: If the UART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bit field can only be written when the UART is disabled (UE=0).

Note: If the UART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the UART is disabled (UE=0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the UART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the UART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE=1 or ORE=1 or NE=1or UDR = 1 in the UART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE=1 or ORE=1 or NE=1 or UDR = 1 (in SPI slave mode) in the UART_ISR register.

41.7.6 UART baud rate register (UART_BRR)

This register can only be written when the UART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]**: UART baud rate

BRR[15:4]

BRR[15:4] = UARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = UARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = UARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

41.7.7 UART guard time and prescaler register (UART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bit field is used to program the Guard time value in terms of number of baud clock periods.

This bit field can only be written when the UART is disabled (UE=0).

Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 7:0 **PSC[7:0]**: Prescaler value

In IrDA low-power and normal IrDA mode:

PSC[7:0] = IrDA Normal and Low-Power baud rate

PSC[7:0] is used to program the prescaler for dividing the UART source clock to achieve the low-power frequency: the source clock is divided by the value given in the register (8 significant bits).

Note: This bit field is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Refer to Section 41.4: UART implementation on page 1510

41.7.8 UART receiver timeout register (UART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block Length

This bit field gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

This bit field can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bit field gives the Receiver timeout value in terms of number of bits during which there is no activity on the RX line.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the Receiver timeout feature is not supported. Refer to Section 41.4: UART implementation on page 1510.

41.7.9 UART request register (UART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing '1' to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in Smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the UART_ISR register. If the UART does not support Smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the UART_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO that is clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the UART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and requests to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF flag in the UART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the UART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

41.7.10 UART interrupt and status register [alternate] (UART_ISR)

Address offset: 0x1C

Reset value: 0x0X80 00C0

X = 0 if FIFO is enabled and Smartcard mode is disabled

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of UART_CR3 register that is the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the UART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in RXFTCFG in UART_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the UART_RDR register. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the UART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG threshold is configured to '101', RXFT flag is set if 16 data are available that is 15 data in the RXFIFO and 1 data in the UART_RDR. Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the UART_TDR has been transmitted correctly out of the shift register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the UART_ICR register or by a write to the UART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (that is a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the UART does not support the Smartcard mode, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 24 **RXFF**: RXFIFO full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the UART_RDR register.

An interrupt is generated if the RXFFIE bit =1 in the UART_CR1 register.

0: RXFIFO not full.

1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO empty

This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the UART_RQR register.

An interrupt is generated if the TXFEIE bit =1 (bit 30) in the UART_CR1 register.

0: TXFIFO not empty.

1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the UART.

It can be used to verify that the UART is ready for reception before entering low-power mode.

Note: If the UART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the UART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the UART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bit field. It is cleared by software, writing a 1 to the WUCF in the UART_ICR register.

An interrupt is generated if WUFIE=1 in the UART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the UART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the UART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the UART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the UART_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

Note: If the UART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the UART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the UART_ICR register.

An interrupt is generated if CMIE=1in the UART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: UART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the UART_RQR register.

Note: If the UART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the UART_CR3 register.

Note: If the UART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into UART_TDR. This flag is reset by setting UDRCF bit in the UART_ICR register.

0: No underrun error

1: underrun error

Note: If the UART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received. The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal to or greater than BLEN + 4.

An interrupt is generated if the EOBIE=1 in the UART_CR2 register.

It is cleared by software, writing 1 to the EOBCF in the UART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the UART_ICR register.

An interrupt is generated if RTOIE=1 in the UART_CR2 register.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the UART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the UART_ICR register.

An interrupt is generated if CTSIE=1 in the UART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the UART_ICR.

An interrupt is generated if LBDIE = 1 in the UART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the UART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the UART_TDR. Every write operation to the UART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data cannot be written into the UART_TDR.

An interrupt is generated if the TXFNFIE bit =1 in the UART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the UART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.

An interrupt is generated if TCIE=1 in the UART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the UART_ICR register or by a write to the UART_TDR register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the UART_RDR register. Every read operation from the UART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the UART_RQR register.

An interrupt is generated if RXFNEIE=1 in the UART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the UART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the UART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the UART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the UART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the UART_ICR register.

An interrupt is generated if RXFNEIE=1 or EIE = 1 in the UART_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the UART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the UART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the UART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the UART tolerance to deviations (Refer to Section 41.5.8: Tolerance of the UART receiver to clock deviation on page 1526).

This error is associated with the character in the UART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the UART_ICR register.

An interrupt is generated if EIE = 1 in the UART_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the UART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the UART_ICR register.

An interrupt is generated if PEIE = 1 in the UART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the UART_RDR.

41.7.11 UART interrupt and status register [alternate] (UART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the UART_TDR has been transmitted correctly out of the shift register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the UART_ICR register or by a write to the UART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (that is a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the UART does not support the Smartcard mode, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the UART.

It can be used to verify that the UART is ready for reception before entering low-power mode.

Note: If the UART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the UART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the UART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bit field. It is cleared by software, writing a 1 to the WUCF in the UART_ICR register.

An interrupt is generated if WUFIE=1 in the UART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the UART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the UART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the UART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the UART_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

Note: If the UART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the UART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the UART_ICR register.

An interrupt is generated if CMIE=1in the UART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: UART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the UART_RQR register.

Note: If the UART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the UART_CR3 register.

Note: If the UART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into UART_TDR. This flag is reset by setting UDRCF bit in the UART_ICR register.

0: No underrun error

1: underrun error

Note: If the UART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received. The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal to or greater than BLEN + 4.

An interrupt is generated if the EOBIE=1 in the UART_CR2 register.

It is cleared by software, writing 1 to the EOBCF in the UART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the UART_ICR register.

An interrupt is generated if RTOIE=1 in the UART_CR2 register.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the UART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the UART_ICR register.

An interrupt is generated if CTSIE=1 in the UART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the UART_ICR.

An interrupt is generated if LBDIE = 1 in the UART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the UART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the UART_TDR register has been transferred into the shift register. It is cleared by writing to the UART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the UART_RQR register, in order to discard the data.

An interrupt is generated if the TXEIE bit =1 in the UART_CR1 register.

0: Data register full

1: Data register not full

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the UART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE=1 in the UART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the UART_ICR register or by a write to the UART_TDR register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the UART_RDR shift register has been transferred to the UART_RDR register. It is cleared by reading from the UART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the UART_RQR register.

An interrupt is generated if RXNEIE=1 in the UART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the UART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the UART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the UART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the UART_RDR register while RXNE=1. It is cleared by a software, writing 1 to the ORECF, in the UART_ICR register.

An interrupt is generated if RXNEIE=1 or EIE = 1 in the UART_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the UART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the UART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the UART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the UART tolerance to deviations (Refer to Section 41.5.8: Tolerance of the UART receiver to clock deviation on page 1526).

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the UART_ICR register.

An interrupt is generated if EIE = 1 in the UART_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the UART_ICR register.

An interrupt is generated if PEIE = 1 in the UART_CR1 register.

0: No parity error

1: Parity error

41.7.12 UART interrupt flag clear register (UART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	W	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the UART_ISR register.

Note: If the UART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the UART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag
Writing 1 to this bit clears the UDRF flag in the UART_ISR register.
Note: If the UART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510

Bit 12 **EOBCF**: End of block clear flag
Writing 1 to this bit clears the EOBF flag in the UART_ISR register.
Note: If the UART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the UART_ISR register.
Note: If the UART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the UART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the UART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 41.4: UART implementation on page 1510.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the UART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the UART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the UART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the UART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the UART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the UART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the UART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the UART_ISR register.

41.7.13 UART receive data register (UART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 663*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

41.7.14 UART transmit data register (UART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The UART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 663*).

When transmitting with the parity enabled (PCE bit set to 1 in the UART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF=1.

41.7.15 UART prescaler register (UART_PRESC)

This register can only be written when the UART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The UART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaining combinations: Reserved

Note: When PRESCALER is programmed with a value different from the allowed ones, programmed prescaler value is 1011 that is input clock divided by 256.

42 Serial peripheral interface / integrated interchip sound (SPI/I2S)

42.1 Introduction

The SPI/I²S interface can be used to communicate with external devices using the SPI protocol or the I^2S audio protocol. SPI or I^2S mode is selectable by software. SPI Motorola mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

The integrated interchip sound (I^2S) protocol is also a synchronous serial communication interface.It can operate in slave or master mode with half-duplex communication. It can address four different audio standards including the Philips I^2S standard, the MSB- and LSB-justified standards and the PCM standard.

42.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4 to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to $f_{PCLK}/2$
- Slave mode frequency up to $f_{PCLK}/2$.
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- Enhanced TI and NSS pulse modes support

42.3 I2S main features

- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode, overrun flag in reception mode (master and slave) and Frame Error Flag in reception and transmitter mode (slave only)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock can be output to drive an external audio component. Ratio is fixed at 256 × F_S (or 128 × F_S in case of PCM mode) (where F_S is the audio sampling frequency)

42.4 SPI/I2S implementation

The following table describes all the SPI instances and their features embedded in the devices.

Table 469. SR5E1x SPI and SPI/I2S implementation

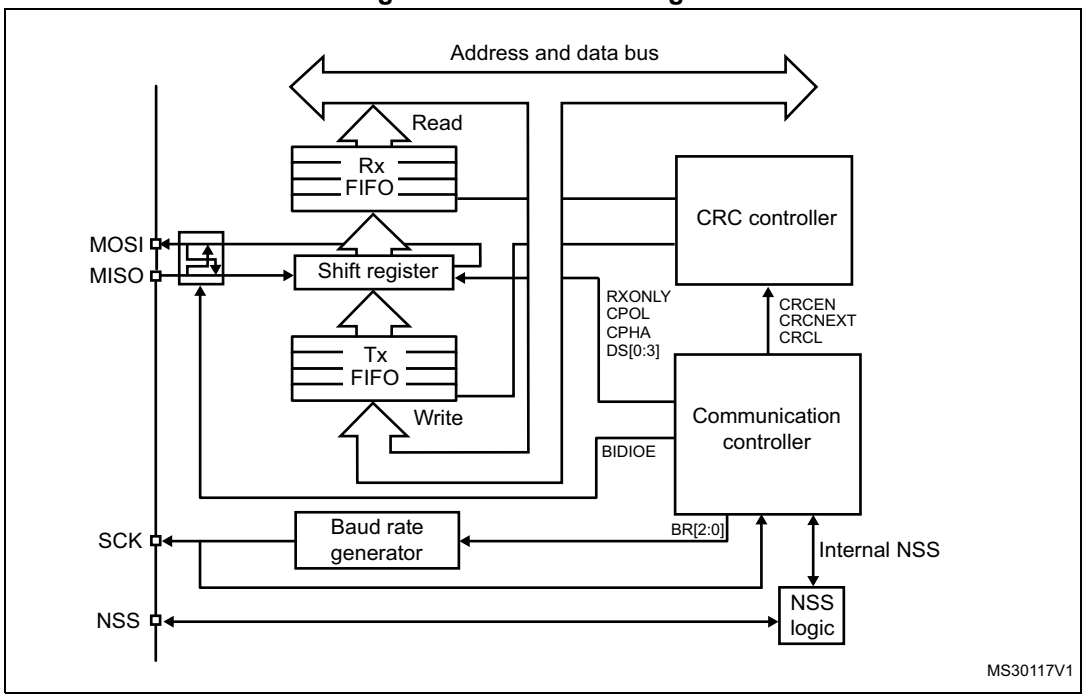
SPI features	SPI1	SPI2S2	SPI2S3	SPI4
Enhanced NSSP & TI modes	Yes	Yes	Yes	Yes
Hardware CRC calculation	Yes	Yes	Yes	Yes
I2S support	No	Yes	Yes	No
Data size configurable	from 4 to 16-bit	from 4 to 16-bit	from 4 to 16-bit	from 4 to 16-bit
Rx/Tx FIFO size	32-bit	32-bit	32-bit	32-bit
Wakeup capability from Low-power Sleep	Yes	Yes	Yes	Yes

42.5 SPI functional description

42.5.1 General description

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram.

Figure 681. SPI block diagram



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

  See *Section 42.5.5: Slave select (NSS) pin management* for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

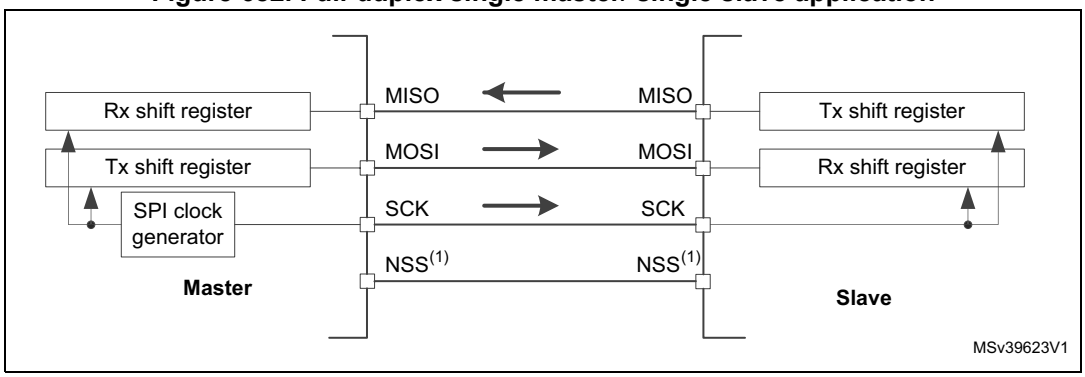
42.5.2 Communications between one master and one slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management). Communication is always initiated by the master.

Full-duplex communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 682. Full-duplex single master/ single slave application

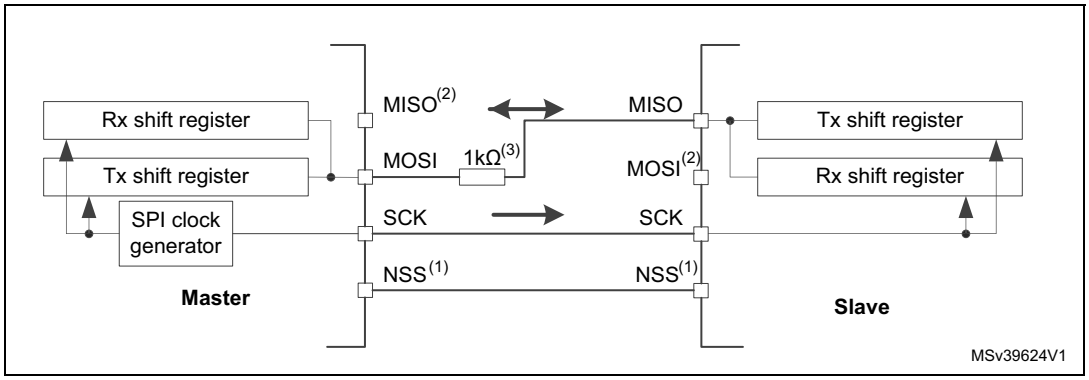


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 42.5.5: Slave select (NSS) pin management*.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx_CR1 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

Figure 683. Half-duplex single master/ single slave application



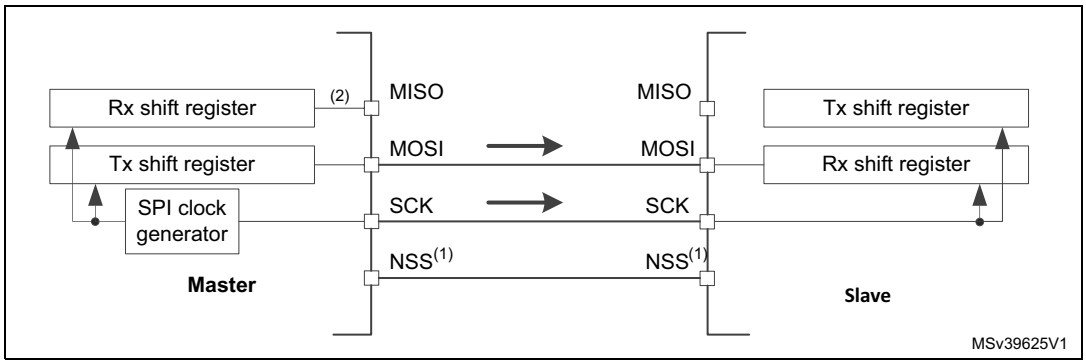
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 42.5.5: Slave select (NSS) pin management*.
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.
3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectionnal mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communication data). Both nodes then fight while providing opposite output levels on the common line temporary till next node changes its direction settings correspondingly, too. It is suggested to insert a serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx_CR2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** the configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1)**: the application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see the *Section 42.5.5: Slave select (NSS) pin management*). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

Figure 684. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)



1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see *Section 42.5.5: Slave select (NSS) pin management*.
2. An accidental input information is captured at the input of transmitter Rx shift register. All the events associated with the transmitter receive flow must be ignored in standard transmit only mode (for example OVF flag).

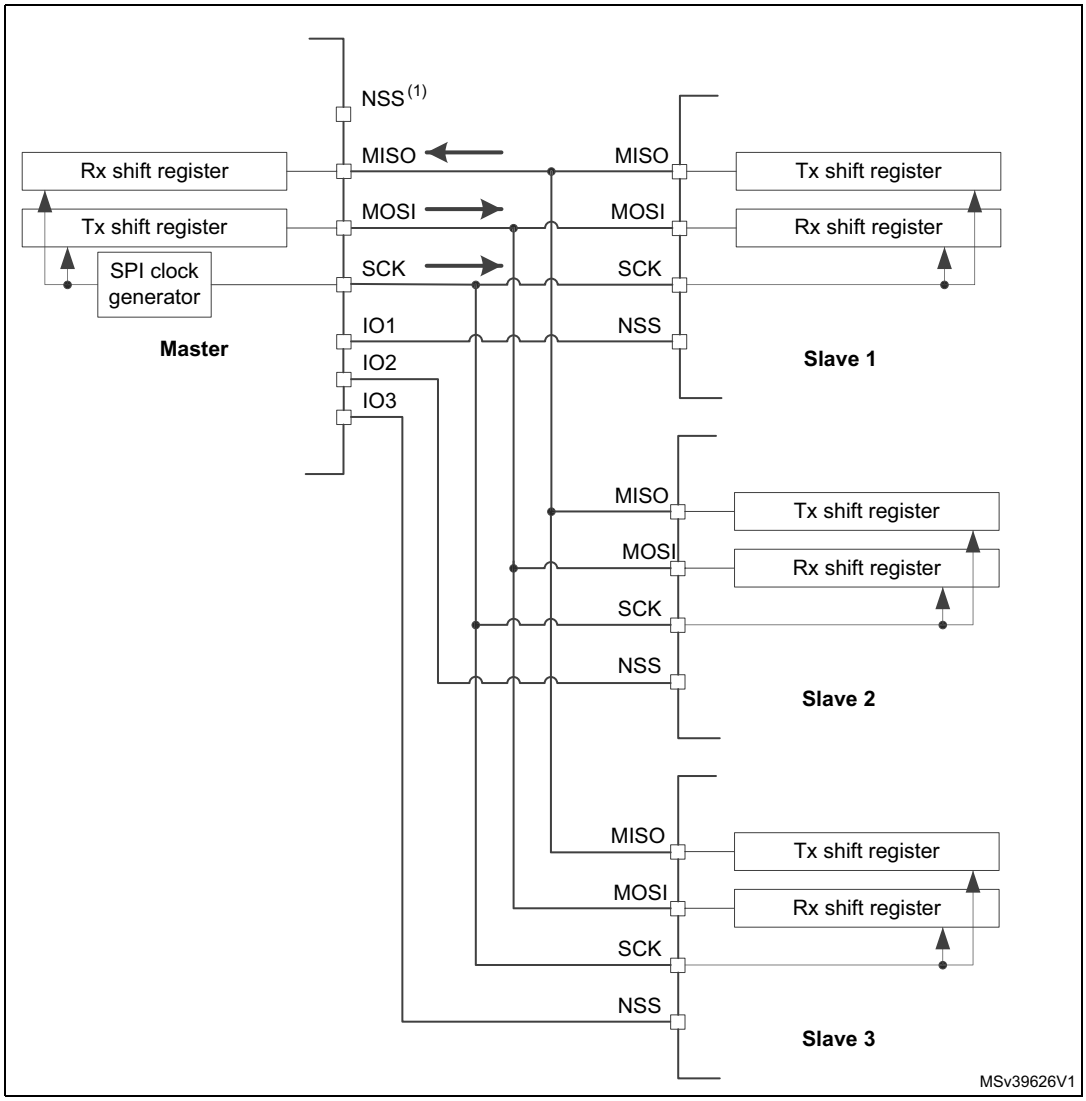
Note: In this configuration, both the MISO pins can be used as GPIOs.

Any simplex communication can be alternatively replaced by a variant of the half-duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).

42.5.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see *Figure 685.*). The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input. When this is done, a standard master and dedicated slave communication is established.

Figure 685. Master and three independent slaves



1. NSS pin is not used on master side at this configuration. It has to be managed internally (SSM=1, SSI=1) to prevent any MODF error.

Note: As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see I/O alternate function input/output section (GPIO)).

42.5.4 Multi-master communication

Unless SPI bus is not designed for a multi-master capability primarily, the user can use built-in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used as configured at hardware input mode.

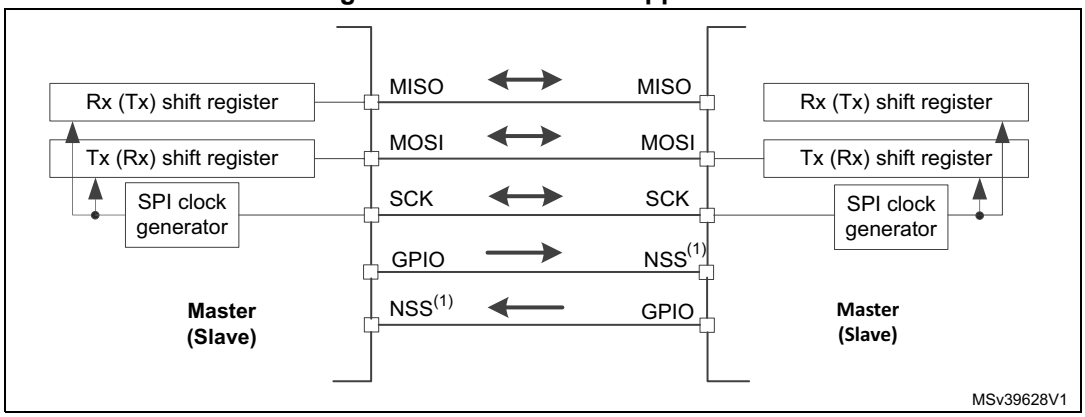
The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via dedicated GPIO pin. After the session is

completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (for example to postpone next attempt by predefined different time-outs applied at both nodes).

Figure 686. Multi-master application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.

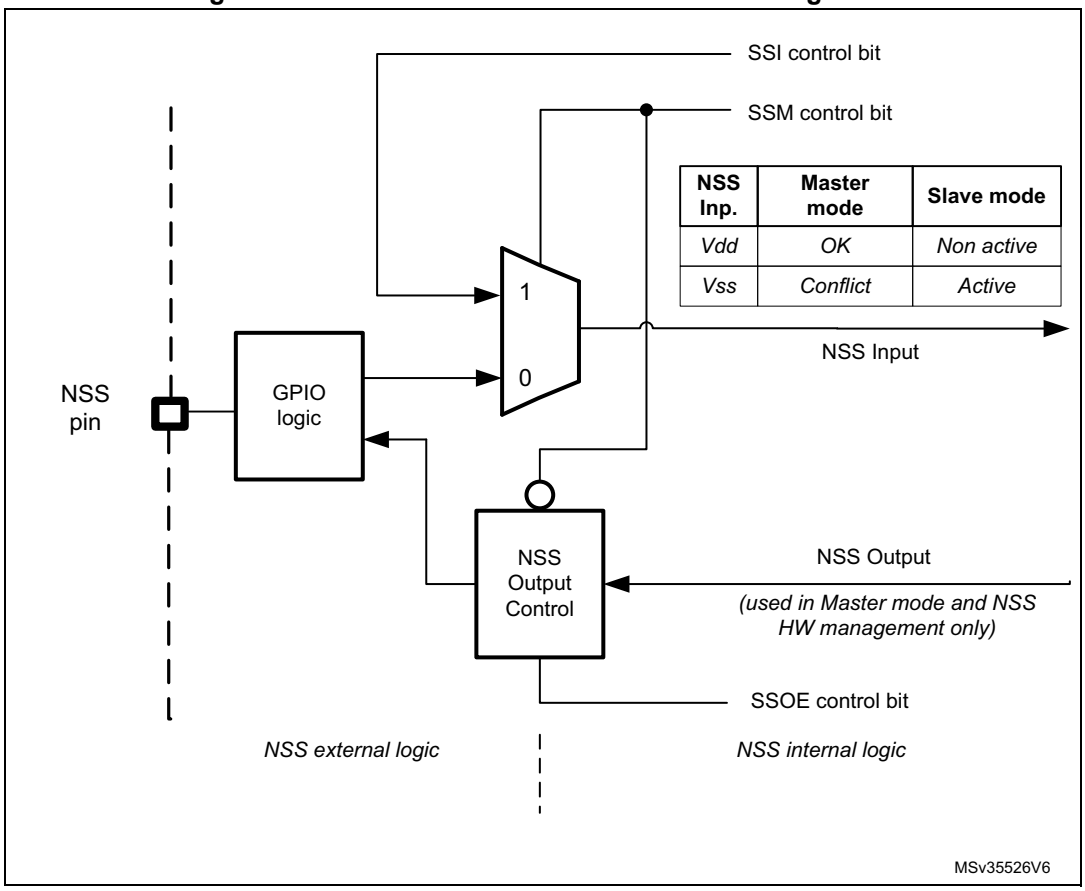
42.5.5 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

- **Software NSS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx_CR2).
  - **NSS output enable (SSM=0,SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
  - **NSS output disable (SSM=0, SSOE = 0)**: if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 687. Hardware/software slave select management



42.5.6 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

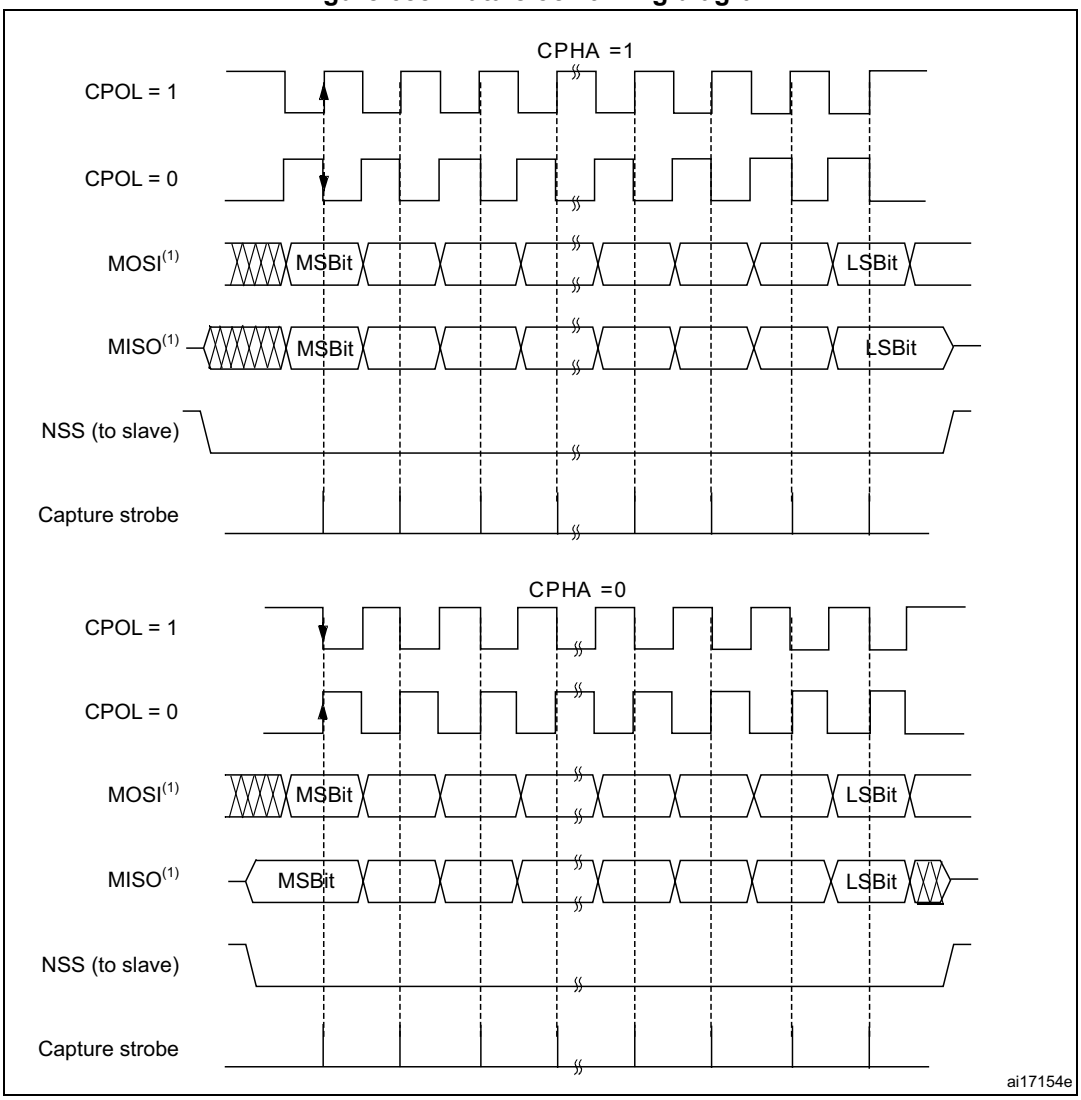
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 688, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPIx_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

Figure 688. Data clock timing diagram

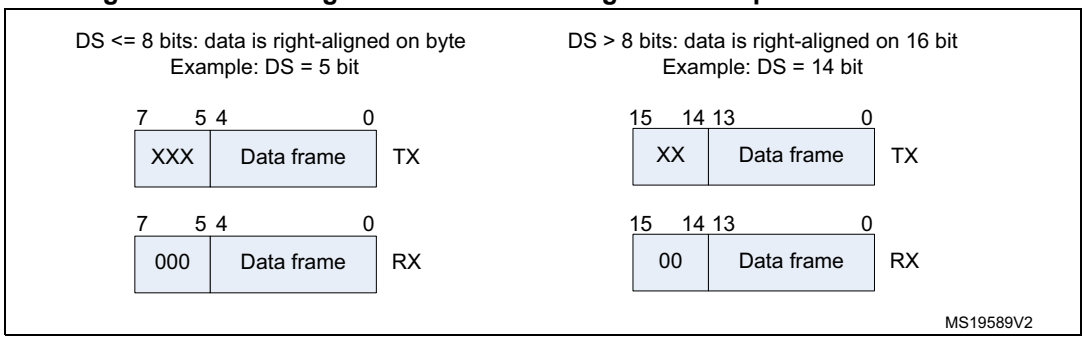


1. The order of data bits depends on LSBFIRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit. The data frame size is chosen by using the DS bits. It can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception. Whatever the selected data frame size, read access to the FIFO must be aligned with the FRXTH level. When the SPIx_DR register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte) or a half-word (see *Figure 689*). During communication, only bits within the data frame are clocked and transferred.

Figure 689. Data alignment when data length is not equal to 8-bit or 16-bit



Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 8-bit data frame size.

42.5.7 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated sections. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI_CR1 register:
   a) Configure the serial clock baud rate using the BR[2:0] bits (Note: 4).
   b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). (Note: 2 - except the case when CRC is enabled at TI mode).
   c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE can't be set at the same time).
   d) Configure the LSBFIRST bit to define the frame format (Note: 2).
   e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
   f) Configure SSM and SSI (Notes: 2 & 3).
   g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI_CR2 register:
   a) Configure the DS[3:0] bits to select the data length for the transfer.
   b) Configure SSOE (Notes: 1 & 2 & 3).
   c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
   d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CHPA and TI bits cleared in NSSP mode).
   e) Configure the FRXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx_DR register.
   f) Initialize LDMA_TX and LDMA_RX bits if DMA is used in packed mode.
4. Write to SPI_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

Note: (1) The step is not required in slave mode.

(2) The step is not required in TI mode.

(3) The step is not required in NSSP mode.

(4) The step is not required in slave mode except slave working at TI mode.

42.5.8 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full-duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

To handle DMA, follow the dedicated section.

42.5.9 Data transmission and reception procedures

RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see *Section 42.5.14: CRC calculation*).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see *Section 42.5.13: TI mode*).

A read access to the SPIx_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, RXFIFO can store up to four data frames, whereas TXFIFO can only store up to three when the data frame format is not greater than 8 bits. This difference prevents possible corruption of 3x 8-bit data frames already stored in the TXFIFO when software tries to write more data in 16-bit mode into TXFIFO. Both TXE and RXNE events can be polled or handled by interrupts. See *Figure 691* through *Figure 694*.

Another way to manage the data exchange is to use DMA (see *Communication using DMA (direct memory addressing)*).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at *Section 42.5.10: SPI status flags*). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half-duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave cannot prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see *Section 42.5.5: Slave select (NSS) pin management*).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full-duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to prevent some dummy byte exchange (refer to *Data packing* section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When

the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional “dummy” data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIiRST bits in the RCC_APBiRSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave.
- When transactions’ streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

Note: If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.

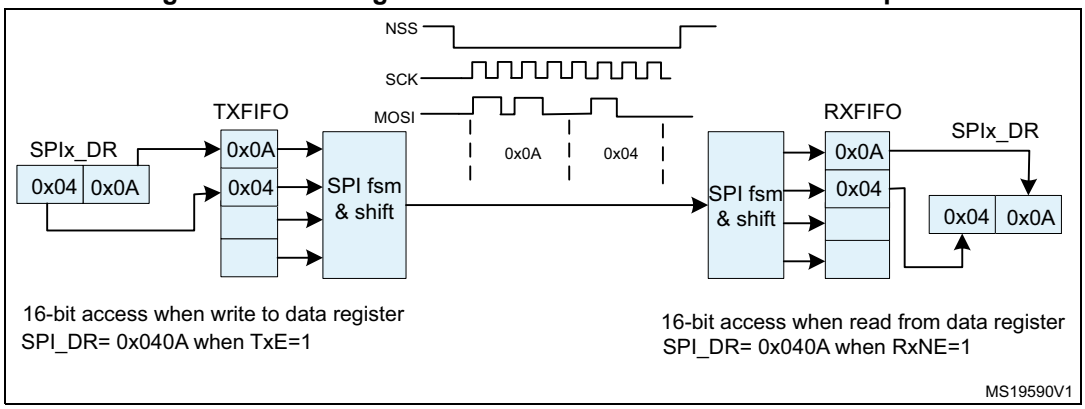
Data packing

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. *Figure 690* provides an example of data packing mode sequence handling. Two data frames are sent after the single 16-bit access the SPIx_DR register of the transmitter. This sequence can generate just one RXNE event in the receiver if the RXFIFO threshold is set to 16 bits (FRXTH=0). The receiver then has to access both data frames by a single 16-bit read of SPIx_DR as a response to this single RXNE event. The

RxFIFO threshold setting and the following read access must be always kept aligned at the receiver side, as data can be lost if it is not in line.

A specific problem appears if an odd number of such “fit into one byte” data frames must be handled. On the transmitter side, writing the last data frame of any odd sequence with an 8-bit access to SPIx_DR is enough. The receiver has to change the Rx_FIFO threshold level for the last data frame received in the odd sequence of frames in order to generate the RXNE event.

Figure 690. Packing data in FIFO for transmission and reception



Note: In this example: Data size DS[3:0] is 4-bit configured, CPOL=0, CPHA=1 and LSBFIRST=0. The Data storage is always right aligned while the valid bits are performed on the bus only, the content of LSB byte goes first on the bus, the unused bits are not taken into account on the transmitter side and padded by zeros at the receiver side.

Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXE or RXNE enable bit in the SPIx_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx_DR register.

See *Figure 691* through *Figure 694*.

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received is not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI. The software must first wait until FTLVL[1:0]=00 and then until BSY=0.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order to:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CR2 register, if DMA Rx is used.
2. Enable DMA streams for Tx and Rx in DMA registers, if the streams are used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CR2 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order to:

1. Disable DMA streams for Tx and Rx in the DMA registers, if the streams are used.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CR2 register, if DMA Tx and/or DMA Rx are used.

Packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPIx_CR2 register) packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel. If the DMA channel PSIZE value is equal to 16-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. The DMA then automatically manages the write operations to the SPIx_DR register.

If data packing mode is used and the number of data to transfer is not a multiple of two, the LDMA_TX/LDMA_RX bits must be set. The SPI then considers only one data for the transmission or reception to serve the last DMA transfer (for more details refer to *Data packing*.)

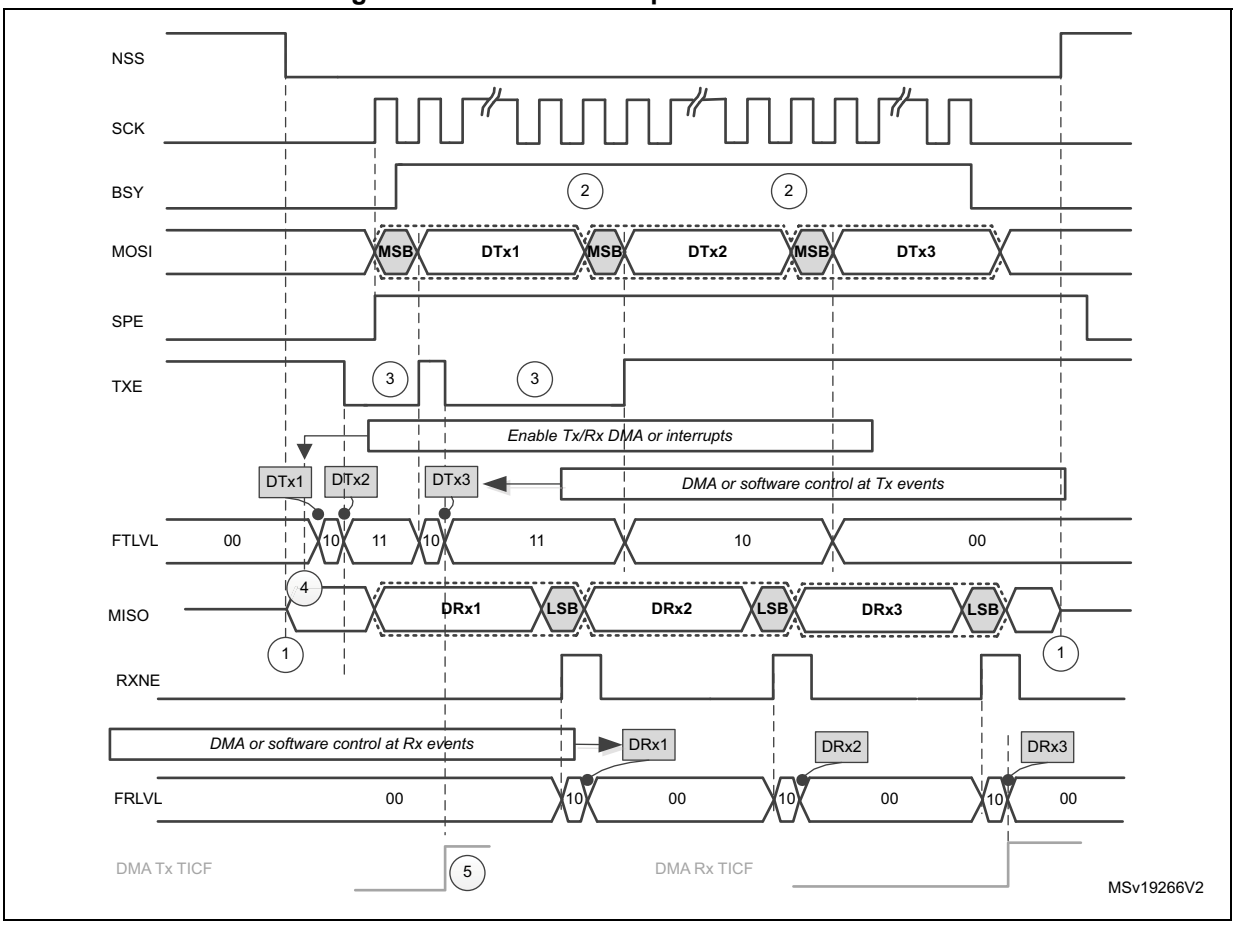
Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by polling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for *Figure 691* through *Figure 694*:

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.
   At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depend on GPIO setting exclusively.
2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.
3. The TXE signal is cleared only if TXFIFO is full.
4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.
5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.
6. The CRC value for a package is calculated continuously frame by frame in the SPIx_TXCRCR and SPIx_RXCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).
   While the CRC value calculated in SPIx_TXCRCR is simply sent out by transmitter, received CRC information is loaded into RxFIFO and then compared with the SPIx_RXCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of RxFIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).
7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the TxFIFO is ¾ full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the TxFIFO becomes ½ full. This frame is stored into TxFIFO with an 8-bit access either by software or automatically by DMA when LDMA_TX control is set.
8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA_RX is set.

Figure 691. Master full-duplex communication



Assumptions for master full-duplex communication example:

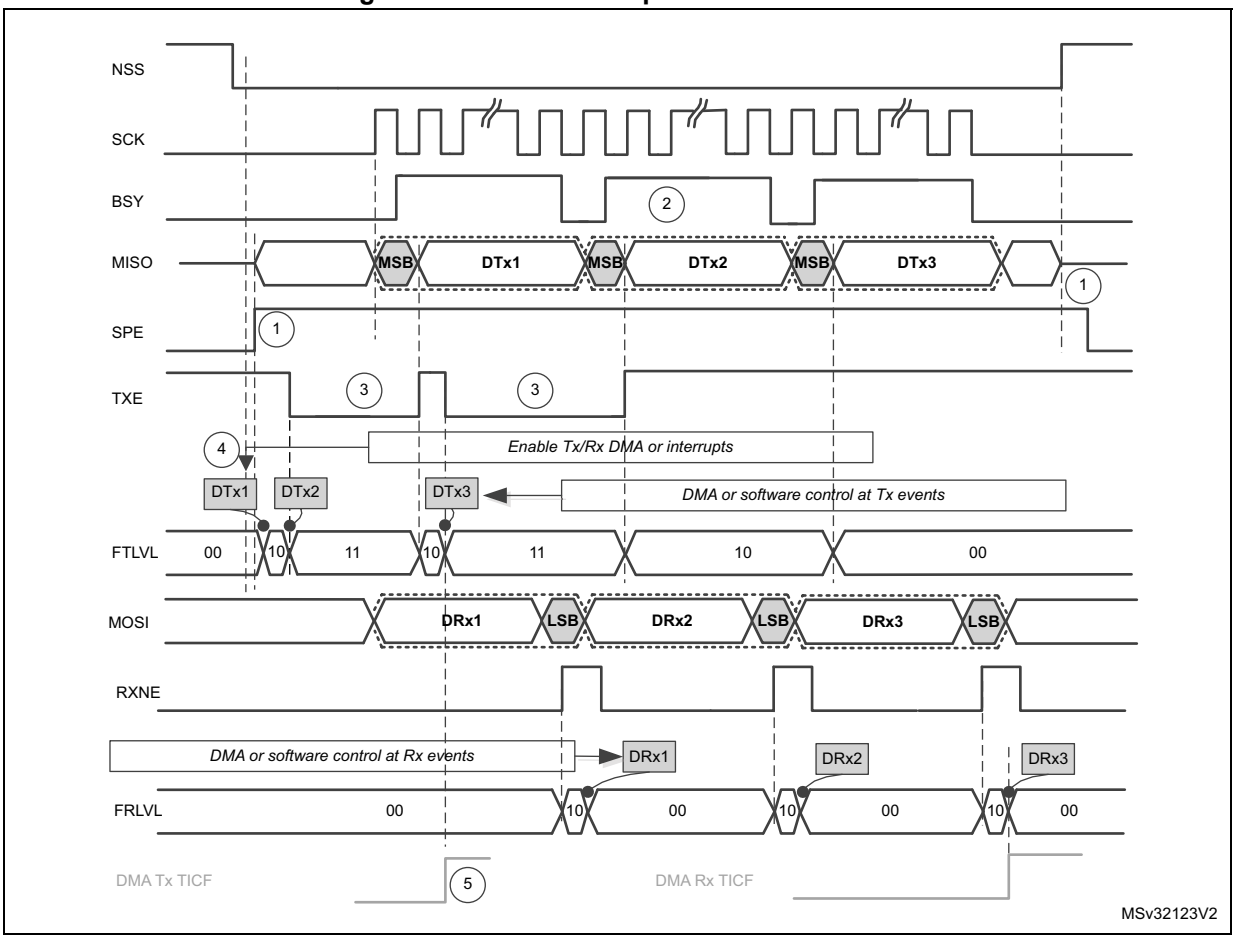
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *Communication diagrams* for details about common assumptions and notes.

Figure 692. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

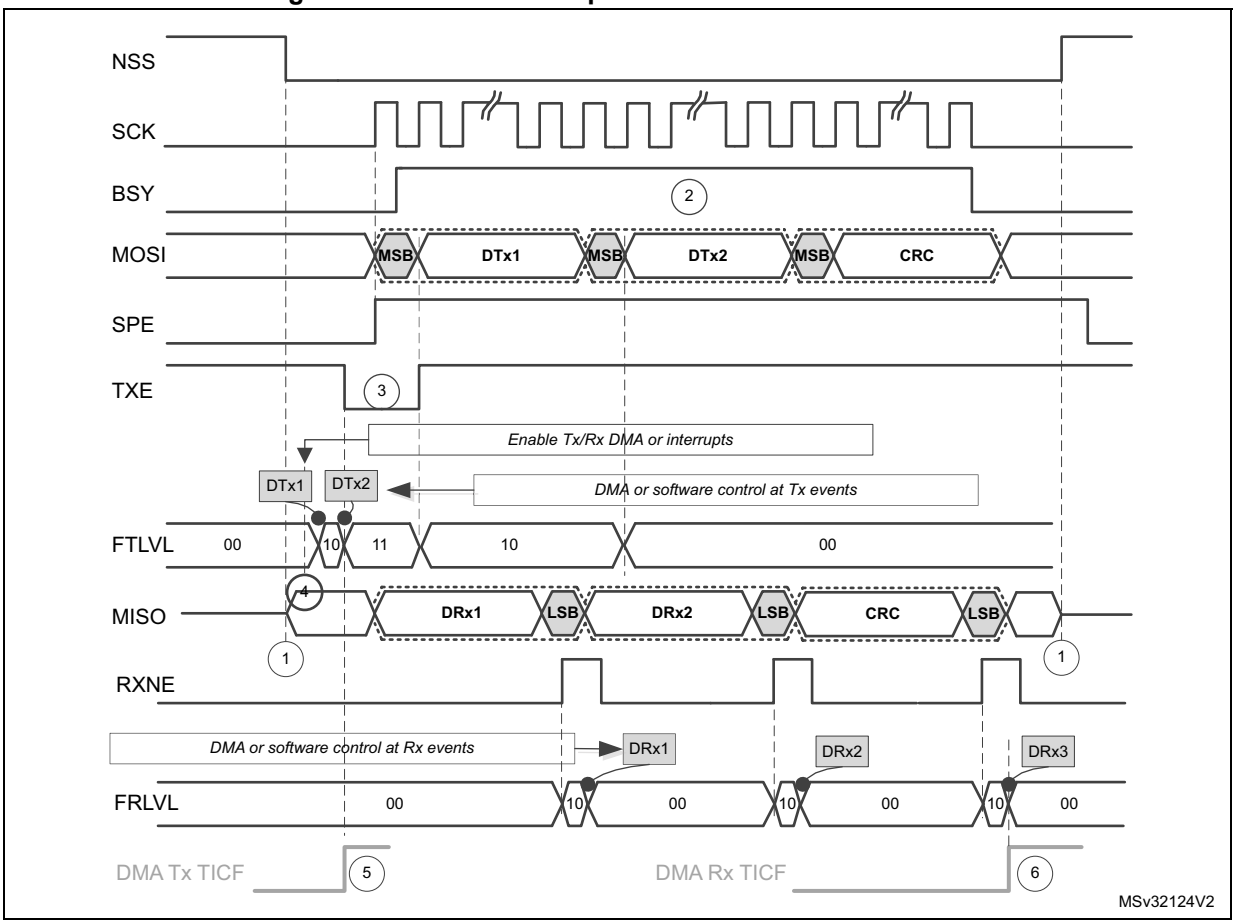
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also *Communication diagrams* for details about common assumptions and notes.

Figure 693. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

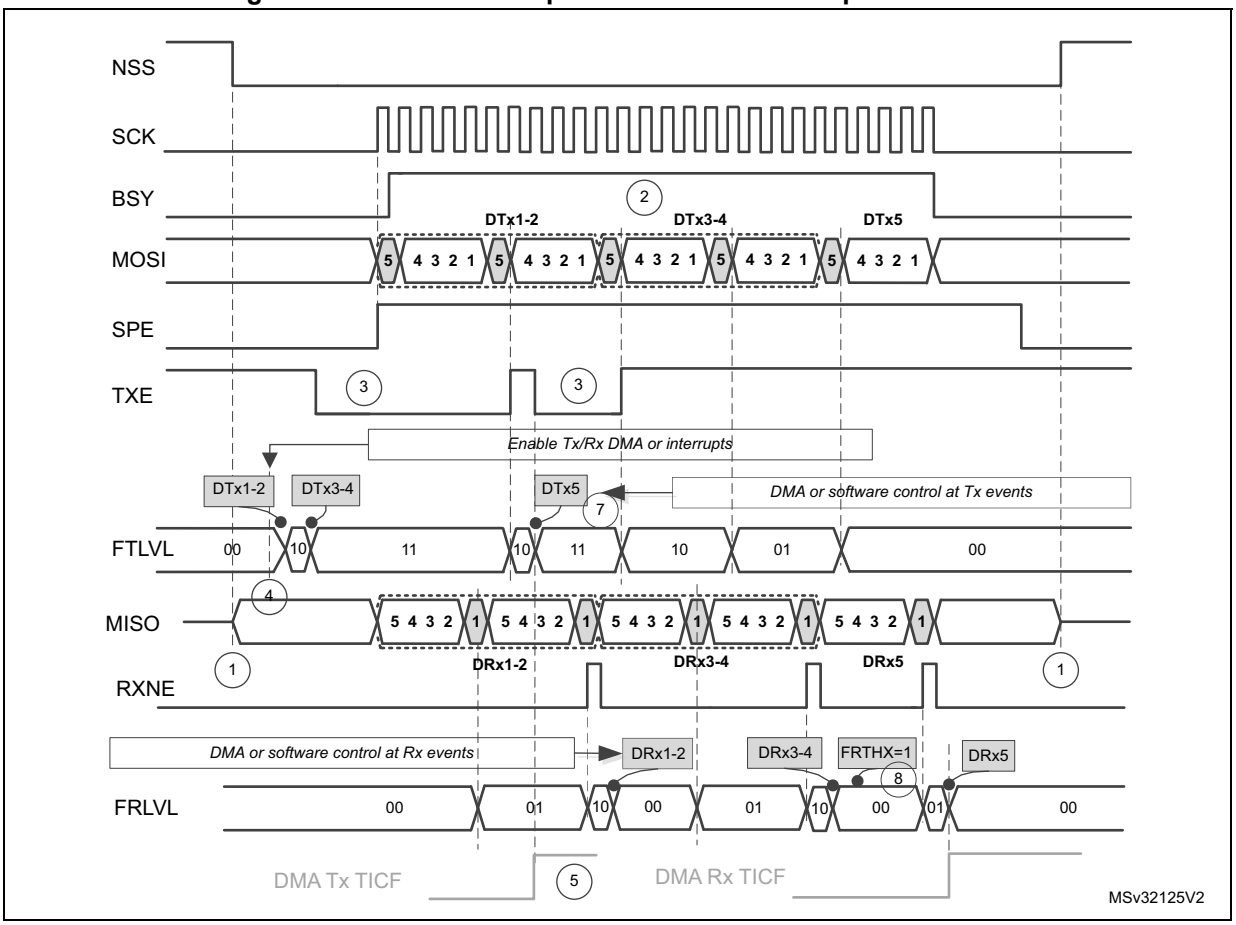
- Data size = 16 bit
- CRC enabled

If DMA is used:

- Number of Tx frames transacted by DMA is set to 2
- Number of Rx frames transacted by DMA is set to 3

See also *Communication diagrams* for details about common assumptions and notes.

Figure 694. Master full-duplex communication in packed mode



Assumptions for master full-duplex communication in packed mode example:

- Data size = 5 bit
- Read/write FIFO is performed mostly by 16-bit access
- FRXTH=0

If DMA is used:

- Number of Tx frames to be transacted by DMA is set to 3
- Number of Rx frames to be transacted by DMA is set to 3
- PSIZE for both Tx and Rx DMA channel is set to 16-bit
- LDMA_TX=1 and LDMA_RX=1

See also *Communication diagrams* for details about common assumptions and notes.

42.5.10 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower than or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx_CR2 register:

- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx_CR2 register is set.

The RXNE is cleared by hardware automatically when the above conditions are no longer true.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled.
- When a fault is detected in Master mode (MODF bit set to 1).
- In Master mode, when it finishes a data transmission and no new data is ready to be sent.
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note: When the next transmission can be handled immediately by the master (for example if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.

42.5.11 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

Overrun flag (OVR)

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited for example the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see *Section 42.5.14: CRC calculation*).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI_DR register followed by a read access to the SPI_SR register.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx_SR register while the MODF bit is set.
2. Then write to the SPIx_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx_CR1 register is set. The CRCERR flag in the SPIx_SR register is set if the value received in the shift register does not match the receiver SPIx_RXCRCR value. The flag is cleared by the software.

TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

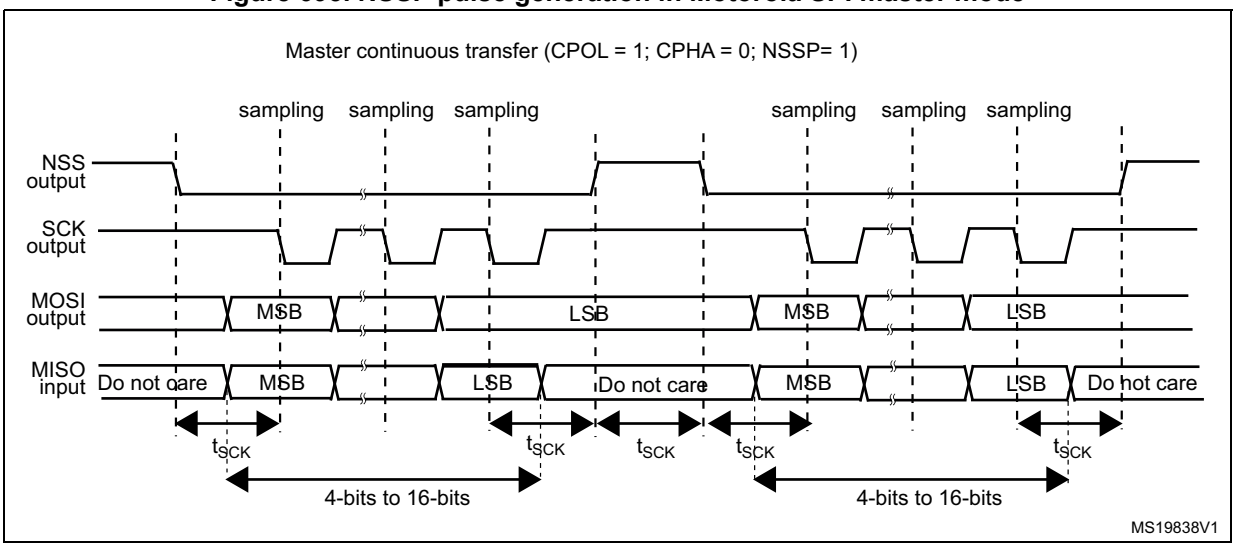
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI must be disabled because data consistency is no longer guaranteed and communications must be reinitiated by the master when the slave SPI is enabled again.

42.5.12 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

Figure 695 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 695. NSSP pulse generation in Motorola SPI master mode



Note: Similar behavior is encountered when CPOL = 0. In this case the sampling edge is the rising edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

42.5.13 TI mode

TI protocol in master mode

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx_CR1 register are. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx_CR1 and SPIx_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see *Figure 696*). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ ($t_{release}$) depends on internal resynchronization and on the

baud rate value set in through the BR[2:0] bits in the SPIx_CR1 register. It is given by the formula:

$$\frac{t_{baud_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud_rate}}{2} + 6 \times t_{pclk}$$

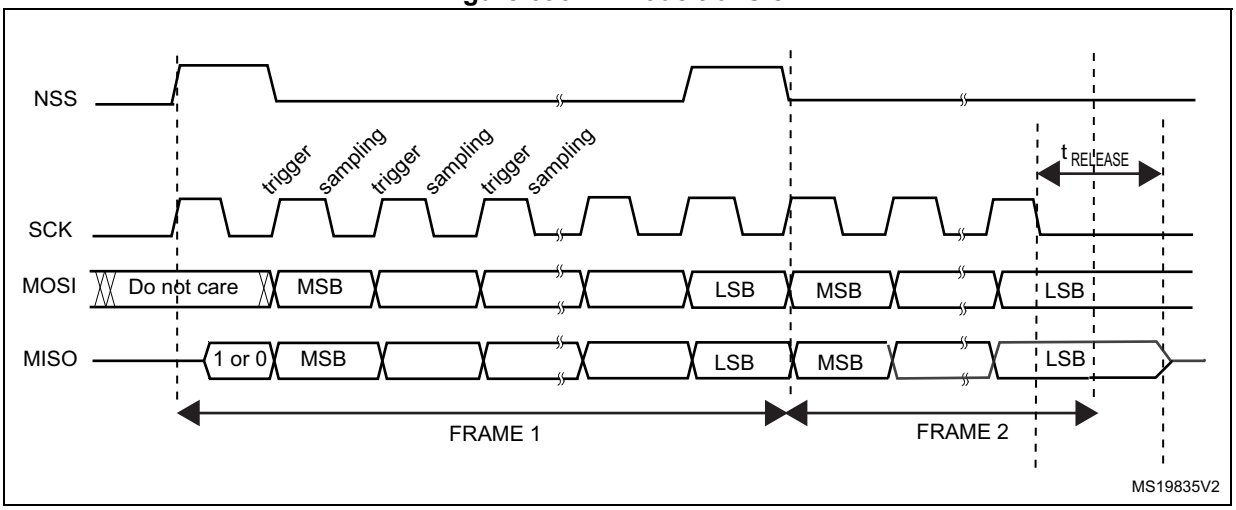
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

If the data size is equal to 4-bit or 5-bit, the master in full-duplex mode or transmit-only mode uses a protocol with one more dummy data bit added after LSB. TI NSS pulse is generated above this dummy bit clock cycle instead of the LSB in each period.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

Figure 696 shows the SPI communication waveforms when TI mode is selected.

Figure 696. TI mode transfer



42.5.14 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

Note: The polynomial value must only be odd. No even values are supported.

CRC transfer managed by CPU

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx_DR register. Then CRCNEXT bit has to be set in the SPIx_CR1 register to indicate that the CRC frame transaction follows after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPIx_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx_DR register in order to clear the RXNE flag.

CRC transfer managed by DMA

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full-duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

DMA_RX = Numb_of_data + 2

In receive only mode, the DMA reception channel counter must contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA_RX bit needs managing if the number of data is odd.

Resetting the SPIx_TXCRC and SPIx_RXCRC values

The SPIx_TXCRC and SPIx_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).

If the SPI is disabled during a communication the following sequence must be followed:

1. Disable the SPI.
2. Clear the CRCEN bit.
3. Enable the CRCEN bit.
4. Enable the SPI.

Note: When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low during transaction of the CRC phase once the CRCNEXT signal is released. That is why the CRC calculation cannot be used at NSS Pulse mode when NSS hardware mode must be applied at slave normally.

At TI mode, despite the fact that clock phase and clock polarity setting is fixed and independent of SPIx_CR1 register, the corresponding setting CPOL=0 CPHA=1 has to be kept at the SPIx_CR1 register anyway if CRC is applied. In addition, the CRC calculation has to be reset between sessions by SPI disable sequence with re-enable the CRCEN bit described above at both master and slave side, else CRC calculation can be corrupted at this specific mode.

42.6 SPI interrupts

During SPI communication an interrupt can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error
- CRC protocol error

Interrupts can be enabled and disabled separately.

Table 470. SPI interrupt requests

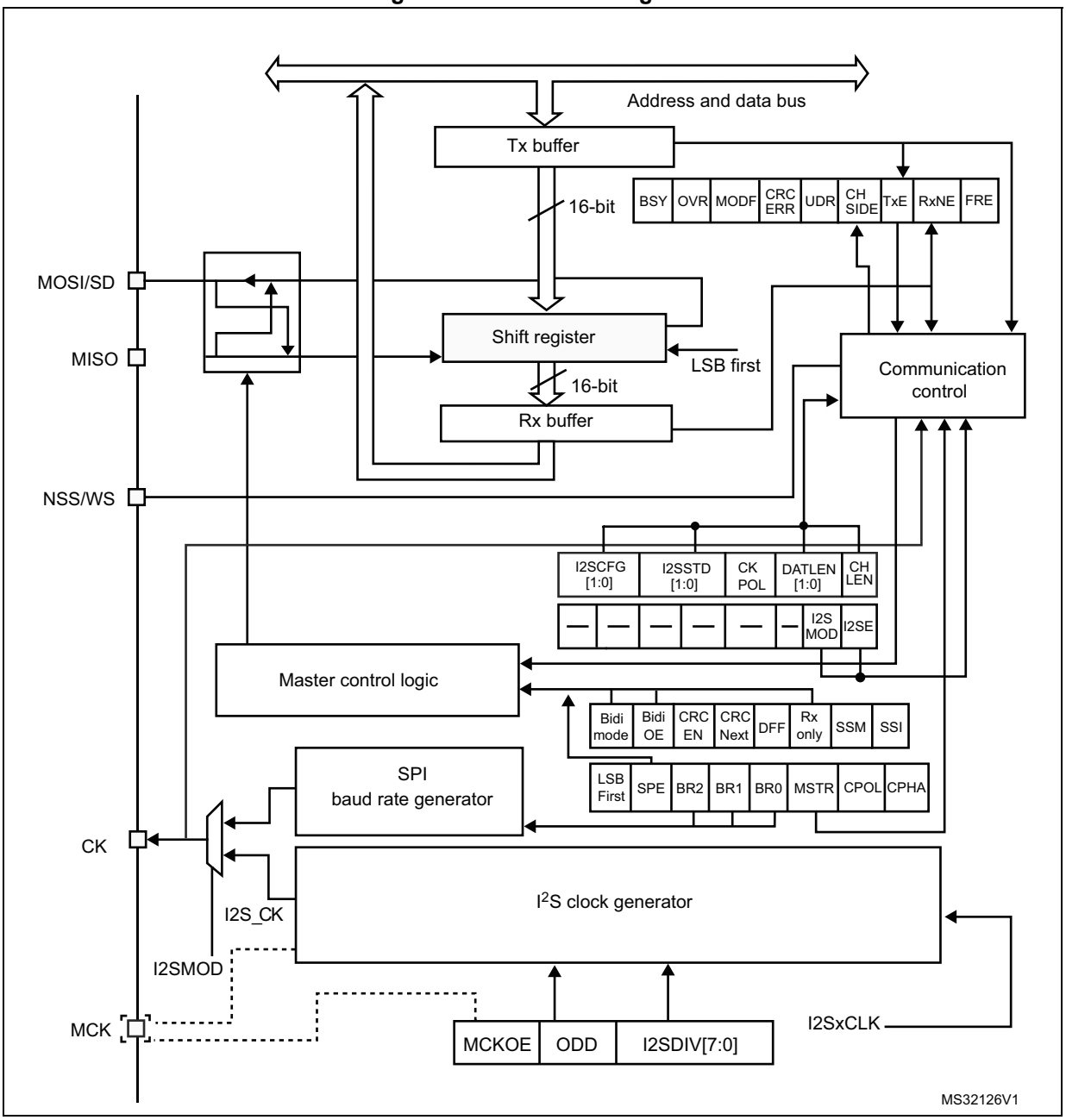
Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	
CRC protocol error	CRCERR	

42.7 I2S functional description

42.7.1 I2S general description

The block diagram of the I2S is shown in *Figure 697*.

Figure 697. I2S block diagram



The SPI can function as an audio I2S interface when the I2S capability is enabled (by setting the I2SMOD bit in the SPIx_I2SCFGR register). This interface mainly uses the same pins, flags and interrupts as the SPI.

The I2S shares three common pins with the SPI:

- SD: Serial Data (mapped on the MOSI pin) to transmit or receive the two time-multiplexed data channels (in half-duplex mode only).
- WS: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I2S is configured in master mode (and when the MCKOE bit in the SPIx_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to 256 × f_S (or 128 fs in case of PCM mode), where f_S is the audio sampling frequency.

The I2S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I^2S mode. One is linked to the clock generator configuration SPIx_I2SPR and the other one is a generic I2S configuration register SPIx_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPIx_CR1 register and all CRC registers are not used in the I^2S mode. Likewise, the SSOE bit in the SPIx_CR2 register and the MODF and CRCERR bits in the SPIx_SR are not used.

The I2S uses the same SPI register for data transfer (SPIx_DR) in 16-bit wide mode.

42.7.2 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for transmission or reception. So, it is up to the software to write into the data register the appropriate value corresponding to each channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPIx_SR register. Channel left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol).

Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in a 16-bit frame
- 16-bit data packed in a 32-bit frame
- 24-bit data packed in a 32-bit frame
- 32-bit data packed in a 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPIx_DR register or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 non-significant bits are extended to 32 bits with 0-bit (by hardware).

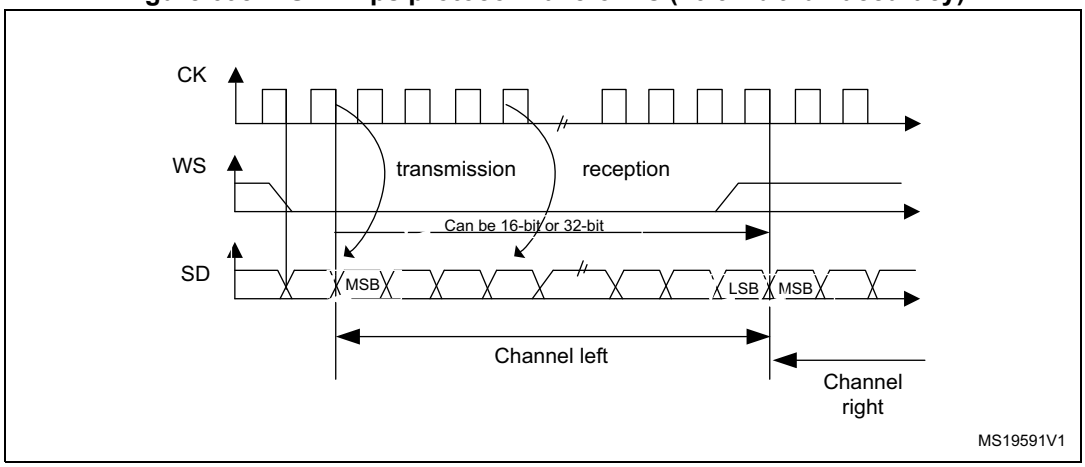
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I^{2}S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx_I2SCFGR register.

I^{2}S Philips standard

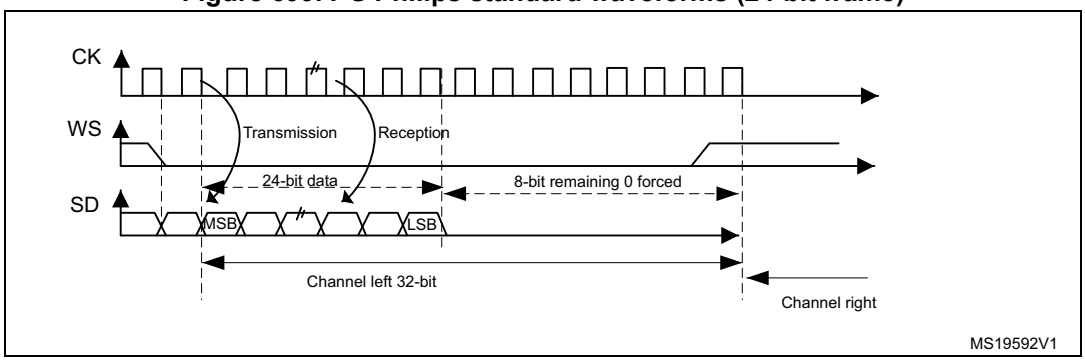
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

Figure 698. I^{2}S Philips protocol waveforms (16/32-bit full accuracy)



Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

Figure 699. I^{2}S Philips standard waveforms (24-bit frame)

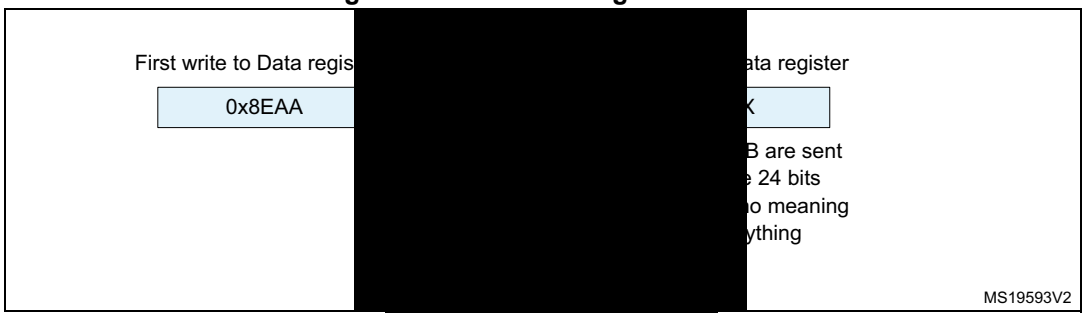


This mode needs two write or read operations to/from the SPIx_DR register.

- In transmission mode:

  If 0x8EAA33 has to be sent (24-bit):

Figure 700. Transmitting 0x8EAA33



- In reception mode:
  If data 0x8EAA33 is received:

Figure 701. Receiving 0x8EAA33

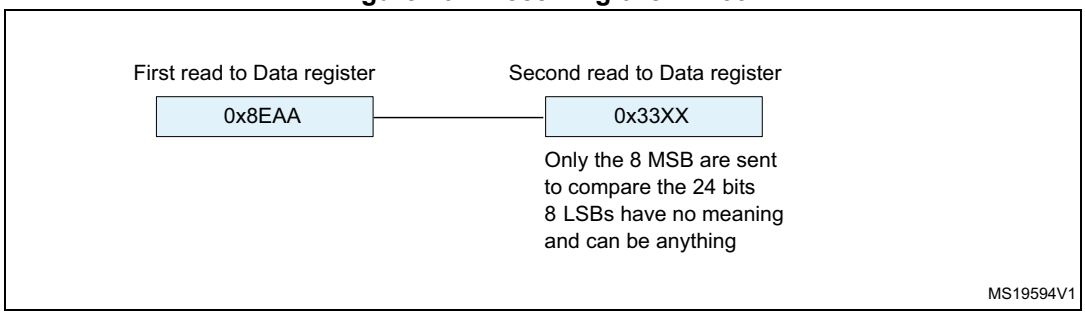
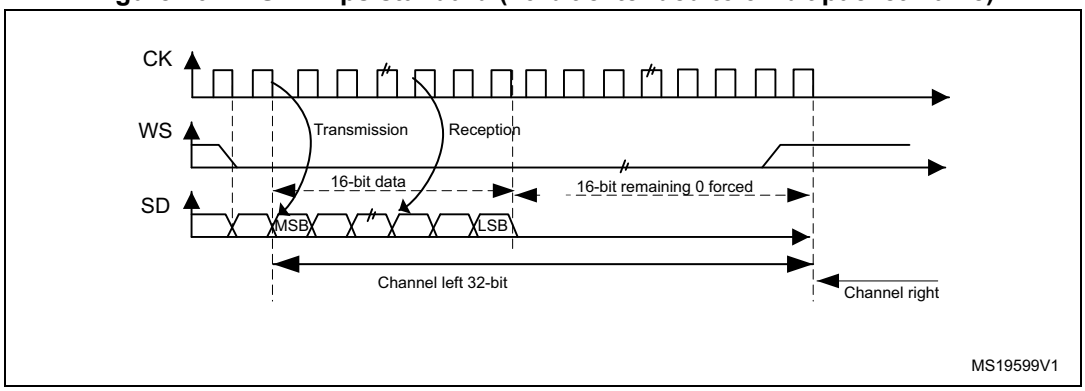


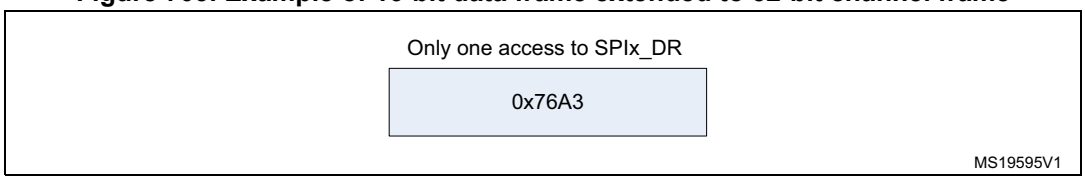
Figure 702. I²S Philips standard (16-bit extended to 32-bit packet frame)



When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in *Figure 703* is required.

Figure 703. Example of 16-bit data frame extended to 32-bit channel frame



For transmission, each time an MSB is written to SPIx_DR, the TXE flag is set and its interrupt, if allowed, is generated to load the SPIx_DR register with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

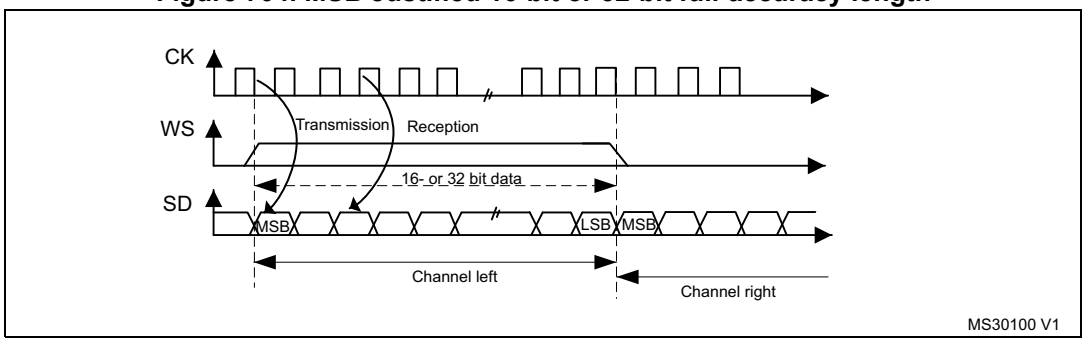
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

MSB justified standard

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

Figure 704. MSB Justified 16-bit or 32-bit full-accuracy length



Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

Figure 705. MSB justified 24-bit frame length

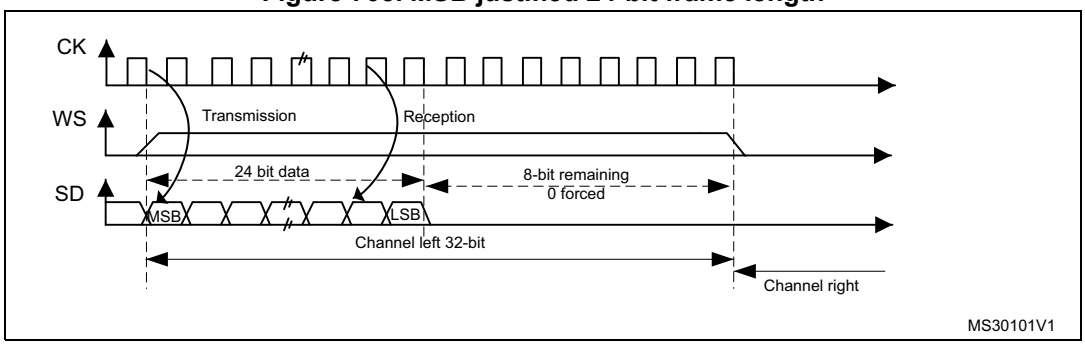
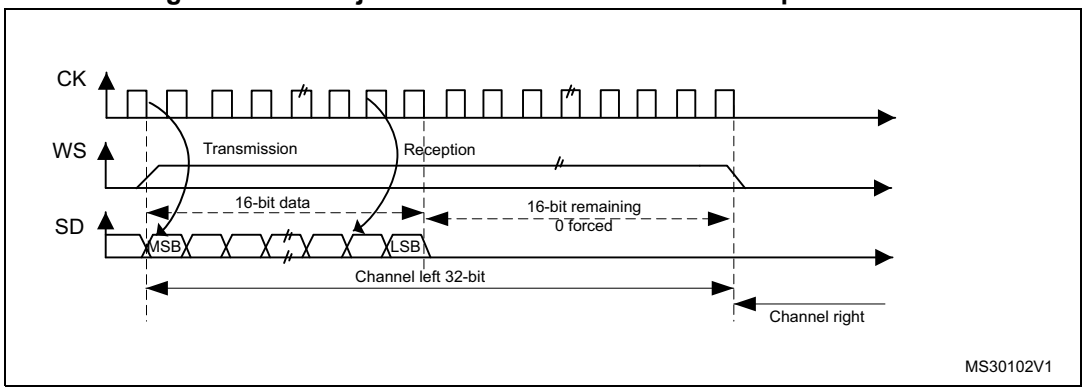


Figure 706. MSB justified 16-bit extended to 32-bit packet frame



LSB justified standard

This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

The sampling of the input and output signals is the same as for the I^2S Philips standard.

Figure 707. LSB justified 16-bit or 32-bit full-accuracy

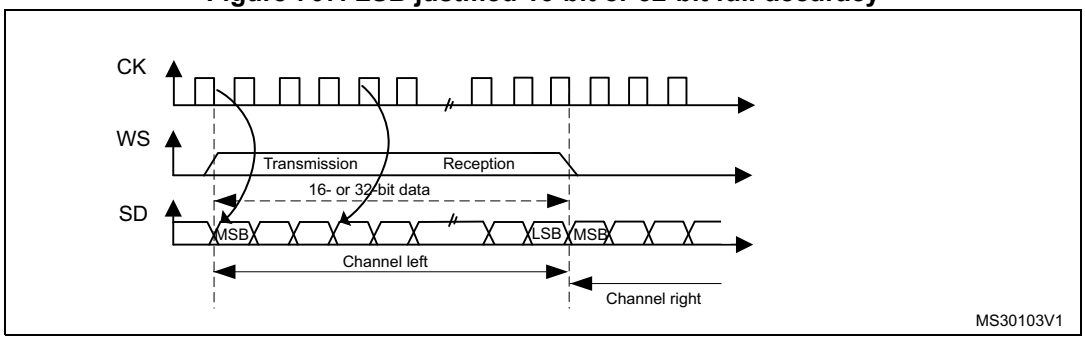
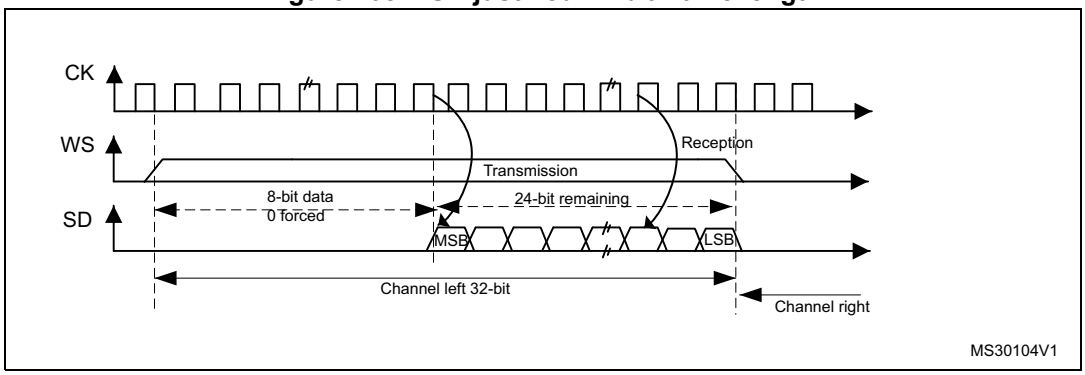


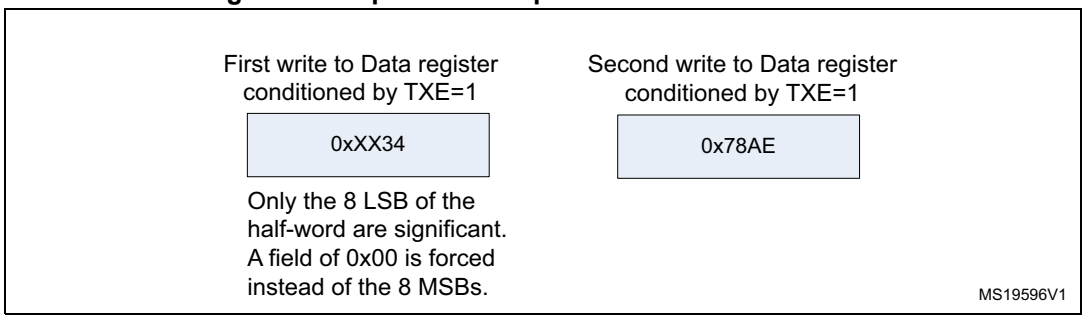
Figure 708. LSB justified 24-bit frame length



- In transmission mode:

  If data 0x3478AE have to be transmitted, two write operations to the SPIx_DR register are required by software or by DMA. The operations are shown below.

Figure 709. Operations required to transmit 0x3478AE



- In reception mode:

  If data 0x3478AE are received, two successive read operations from the SPIx_DR register are required on each RXNE event.

Figure 710. Operations required to receive 0x3478AE

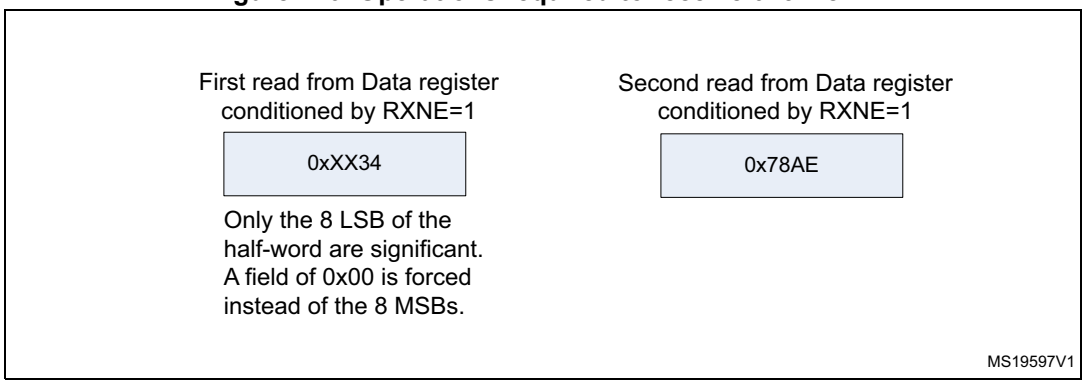
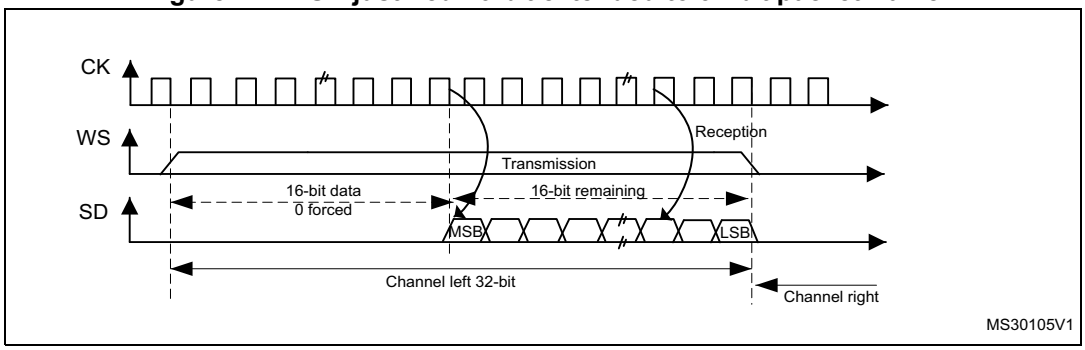


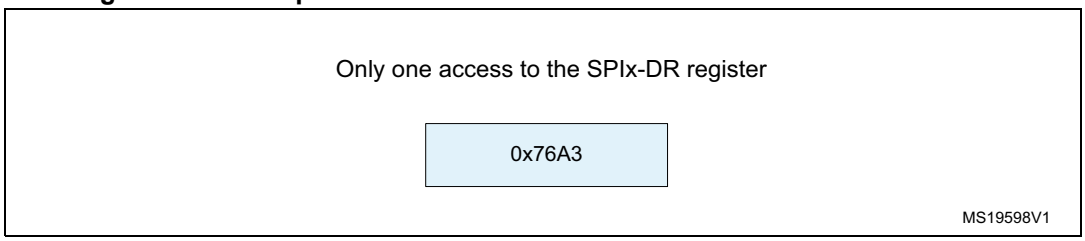
Figure 711. LSB justified 16-bit extended to 32-bit packet frame



When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, Only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in *Figure 712* is required.

Figure 712. Example of 16-bit data frame extended to 32-bit channel frame



In transmission mode, when a TXE event occurs, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). The TXE flag is set again as soon as the effective data (0x76A3) is sent on SD.

In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

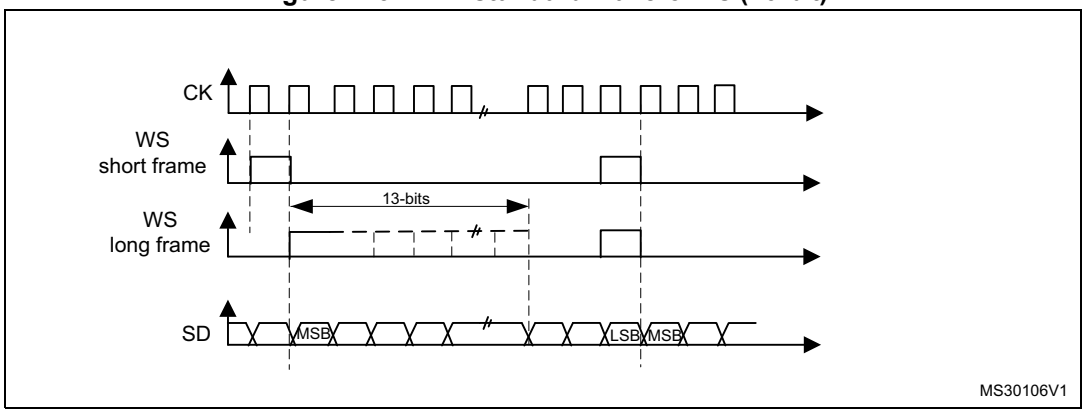
PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPIx_I2SCFGR register.

In PCM mode, the output signals (WS, SD) are sampled on the rising edge of CK signal. The input signals (WS, SD) are captured on the falling edge of CK.

Note that CK and WS are configured as output in MASTER mode.

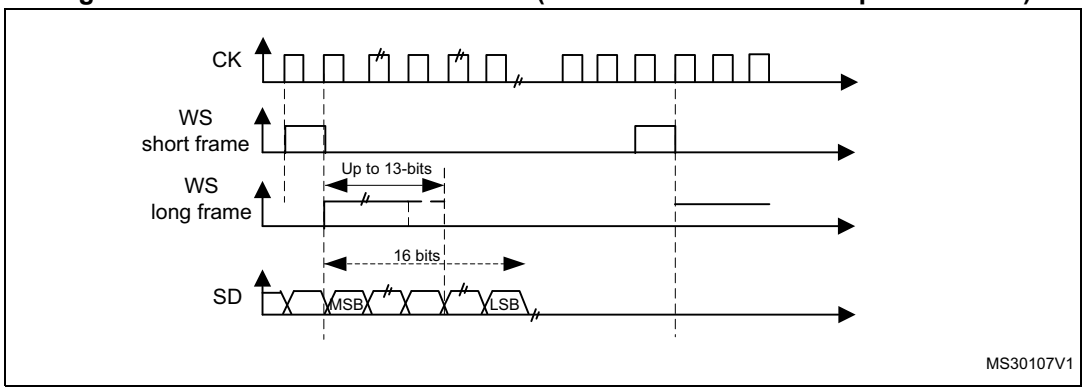
Figure 713. PCM standard waveforms (16-bit)



For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

For short frame synchronization, the WS synchronization signal is only one cycle long.

Figure 714. PCM standard waveforms (16-bit extended to 32-bit packet frame)

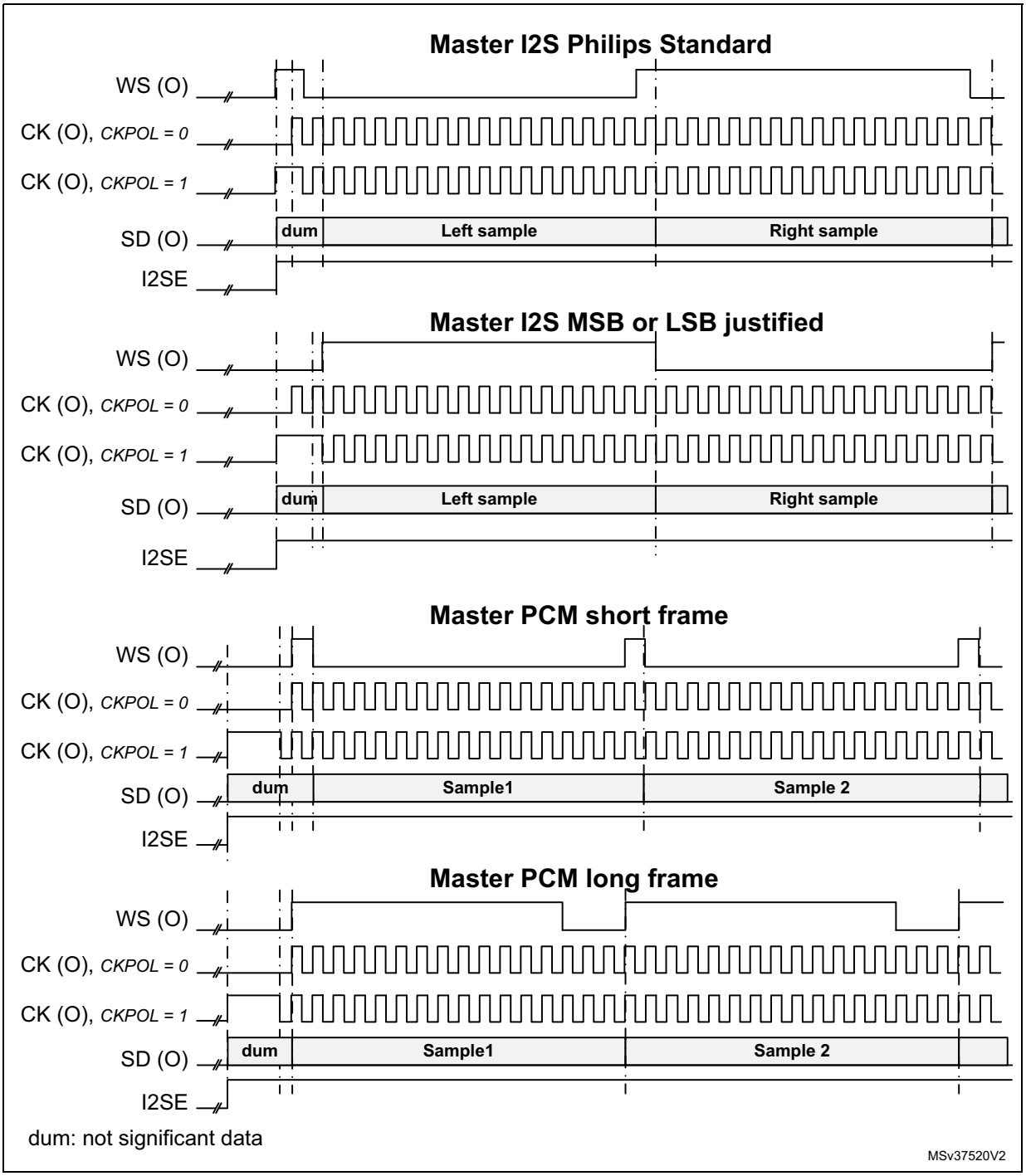


Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPIx_I2SCFGR register) even in slave mode.

42.7.3 Startup description

Figure 715 shows how the serial interface is handled in MASTER mode, when the SPI/I2S is enabled (via I2SE bit). It shows as well the effect of CKPOL on the generated signals.

Figure 715. Start sequence in master mode



In slave mode, the way the frame synchronization is detected, depends on the value of ASTRTEN bit.

If ASTRTEN = 0, when the audio interface is enabled (I2SE = 1), then the hardware waits for the appropriate transition on the incoming WS signal, using the CK signal.

The appropriate transition is a falling edge on WS signal when I²S Philips Standard is used, or a rising edge for other standards. The falling edge is detected by sampling first WS to 1 and then to 0, and vice-versa for the rising edge detection.

If ASTRTEN = 1, the user has to enable the audio interface before the WS becomes active. This means that the I2SE bit must be set to 1 when WS = 1 for I²S Philips standard, or when WS = 0 for other standards.

42.7.4 Clock generator

The I²S bit rate determines the data flow on the I²S data line and the I²S clock signal frequency.

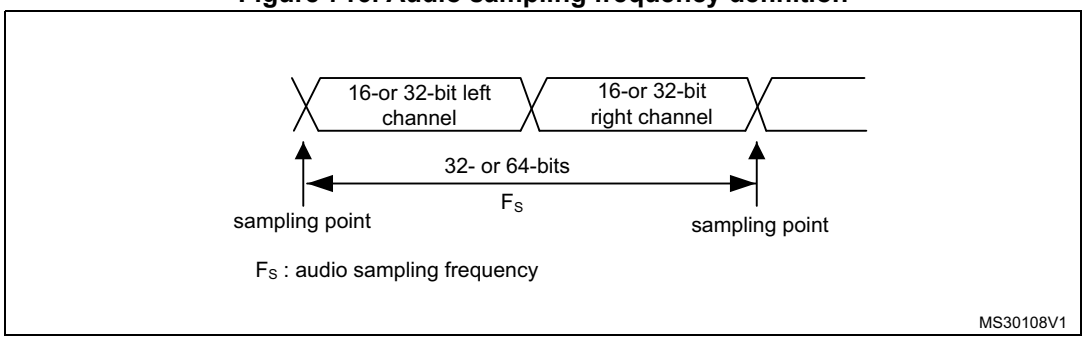
I²S bit rate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I²S bit rate is calculated as follows:

$I^2S \text{ bit rate} = 16 \times 2 \times f_S$

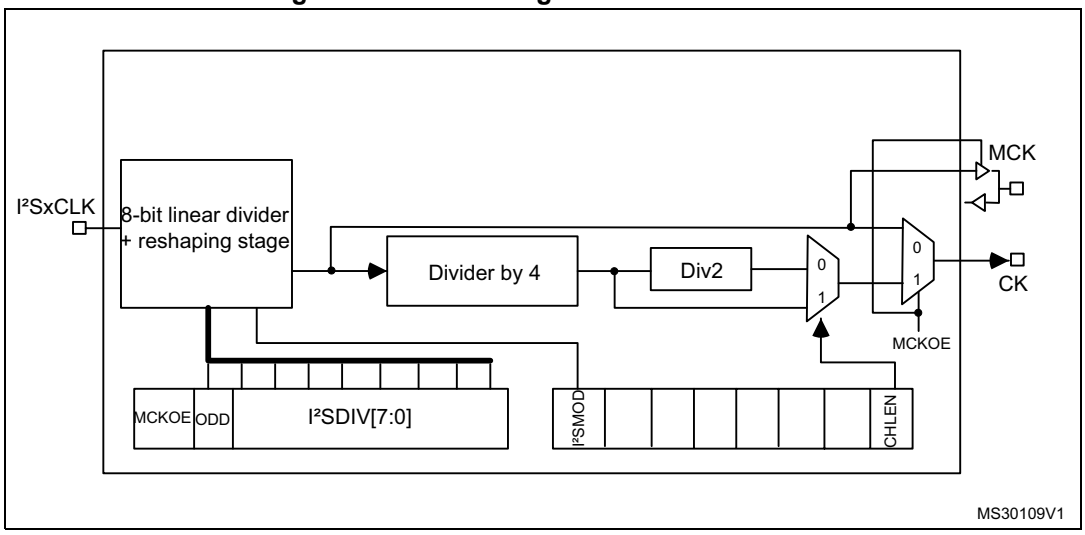
It is: $I^2S \text{ bit rate} = 32 \times 2 \times f_S$ if the packet length is 32-bit wide.

Figure 716. Audio sampling frequency definition



When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency.

Figure 717. I²S clock generator architecture



Note: Where x can be 2 or 3.

Figure 717 presents the communication clock architecture. The I2SxCLK clock is provided by the reset and clock controller (RCC) of the product. The I2SxCLK clock can be asynchronous with respect to the SPI/I2S APB clock.

Warning: In addition, it is mandatory to keep the I2SxCLK frequency higher than or equal to the APB clock used by the SPI/I2S block. If this condition is not respected, the SPI/I2S does not work properly.

The audio sampling frequency may be 192 kHz, 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range).

In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

For I²S modes:

When the master clock is generated (MCKOE in the SPIx_I2SPR register is set):

$$Fs = \frac{F_{I2SxCLK}}{256 \times ((2 \times I2SDIV) + ODD)}$$

When the master clock is disabled (MCKOE bit cleared):

$$Fs = \frac{F_{I2SxCLK}}{32 \times (CHLEN + 1) \times ((2 \times I2SDIV) + ODD)}$$

CHLEN = 0 when the channel frame is 16-bit wide and,

CHLEN = 1 when the channel frame is 32-bit wide.

For PCM modes:

When the master clock is generated (MCKOE in the SPIx_I2SPR register is set):

$$Fs = \frac{F_{I2SxCLK}}{128 \times ((2 \times I2SDIV) + ODD)}$$

When the master clock is disabled (MCKOE bit cleared):

$$Fs = \frac{F_{I2SxCLK}}{16 \times (CHLEN + 1) \times ((2 \times I2SDIV) + ODD)}$$

CHLEN = 0 when the channel frame is 16-bit wide and,

CHLEN = 1 when the channel frame is 32-bit wide.

Where F_S is the audio sampling frequency, and $F_{I2SxCLK}$ is the frequency of the kernel clock provided to the SPI/I2S block.

Note: Note that I2SDIV must be strictly higher than 1.

Table 471 provides example precision values for different clock configurations.

Note: Other configurations are possible that allow optimum clock precision.

Table 471. Audio-frequency precision using standard 8 MHz HSE(1)

SYSCLK (MHz)	Data length	I2SDIV	I2SODD	MCLK	Target fs (Hz)	Real fs (kHz)	Error
48	16	8	0	No	96000	93750	2.3438%
48	32	4	0	No	96000	93750	2.3438%
48	16	15	1	No	48000	48387.0968	0.8065%
48	32	8	0	No	48000	46875	2.3438%
48	16	17	0	No	44100	44117.647	0.0400%
48	32	8	1	No	44100	44117.647	0.0400%
48	16	23	1	No	32000	31914.8936	0.2660%
48	32	11	1	No	32000	32608.696	1.9022%
48	16	34	0	No	22050	22058.8235	0.0400%
48	32	17	0	No	22050	22058.8235	0.0400%
48	16	47	0	No	16000	15957.4468	0.2660%
48	32	23	1	No	16000	15957.447	0.2660%
48	16	68	0	No	11025	11029.4118	0.0400%
48	32	34	0	No	11025	11029.412	0.0400%
48	16	94	0	No	8000	7978.7234	0.2660%
48	32	47	0	No	8000	7978.7234	0.2660%
48	16	2	0	Yes	48000	46875	2.3430%
48	32	2	0	Yes	48000	46875	2.3430%
48	16	2	0	Yes	44100	46875	6.2925%
48	32	2	0	Yes	44100	46875	6.2925%
48	16	3	0	Yes	32000	31250	2.3438%
48	32	3	0	Yes	32000	31250	2.3438%
48	16	4	1	Yes	22050	20833.333	5.5178%
48	32	4	1	Yes	22050	20833.333	5.5178%
48	16	6	0	Yes	16000	15625	2.3438%
48	32	6	0	Yes	16000	15625	2.3438%
48	16	8	1	Yes	11025	11029.4118	0.0400%
48	32	8	1	Yes	11025	11029.4118	0.0400%
48	16	11	1	Yes	8000	8152.17391	1.9022%
48	32	11	1	Yes	8000	8152.17391	1.9022%

1. This table gives only example values for different clock configurations. Other configurations allowing optimum clock precision are possible.

42.7.5 I²S master mode

The I2S can be configured in master mode. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, controlled by the MCKOE bit in the SPIx_I2SPR register.

Procedure

1. Select the I2SDIV[7:0] bits in the SPIx_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPIx_I2SPR register also has to be defined.
2. Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPIx_I2SPR register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the I2SDIV and ODD values must be computed depending on the state of the MCK output, for more details refer to *Section 42.7.4: Clock generator*).
3. Set the I2SMOD bit in the SPIx_I2SCFGR register to activate the I2S functions and choose the I²S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the CHLEN bit. Select also the I²S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.
4. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
5. The I2SE bit in SPIx_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPIx_I2SPR is set.

Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Let's assume the first data written into the Tx buffer corresponds to the left channel data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the right channel have to be written into the Tx buffer. The CHSIDE flag indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high.

A full frame has to be considered as a left channel data transmission followed by a right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

For more details about the write operations depending on the I²S standard mode selected, refer to *Section 42.7.2: Supported audio protocols*.

To ensure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission.

To switch off the I2S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for transmission mode except for the point 3 (refer to the procedure described in *Section 42.7.5: I²S master mode*), where the configuration must set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPIx_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I2S cell.

For more details about the read operations depending on the I²S standard mode selected, refer to *Section 42.7.2: Supported audio protocols*.

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S, specific actions are required to ensure that the I2S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB justified mode (I2SSTD = 10)
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait 17 I2S clock cycles (using a software loop)
  c) Disable the I2S (I2SE = 0)
- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, I²S or PCM modes (I2SSTD = 00, I2SSTD = 01 or I2SSTD = 11, respectively)
  a) Wait for the last RXNE
  b) Then wait 1 I2S clock cycle (using a software loop)
  c) Disable the I2S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I2S:
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait one I2S clock cycle (using a software loop)
  c) Disable the I2S (I2SE = 0)

Note: The BSY flag is kept low during transfers.

42.7.6 I²S slave mode

For the slave configuration, the I2S can be configured in transmission or reception mode. The operating mode is following mainly the same rules as described for the I²S master configuration. In slave mode, there is no clock to be generated by the I2S interface. The

clock and WS signals are input from the external master connected to the I2S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPIx_I2SCFGR register to select I^2S mode and choose the I^2S standard through the I2SSTD[1:0] bits, the data length through the DATLEN[1:0] bits and the number of bits per channel for the frame configuring the CHLEN bit. Select also the mode (transmission or reception) for the slave through the I2SCFG[1:0] bits in SPIx_I2SCFGR register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
3. The I2SE bit in SPIx_I2SCFGR register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I2S data register has to be loaded before the master initiates the communication.

For the I2S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I2S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

Note: The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

Note that the TXE flag must be checked to be at 1 before attempting to write the Tx buffer.

For more details about the write operations depending on the I^2S standard mode selected, refer to *Section 42.7.2: Supported audio protocols*.

To secure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPIx_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPIx_CR2 register, an interrupt is generated when the UDR flag in the SPIx_SR register goes high. In this case, it is mandatory to switch off the I2S and to restart a data transfer starting from the left channel.

To switch off the I2S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in *Section 42.7.6: I²S slave mode*), where the configuration must set the master reception mode using the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.

Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPIx_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPIx_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer.

The CHSIDE flag is updated each time data are received to be read from the SPIx_DR register. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

For more details about the read operations depending on the I²S standard mode selected, refer to *Section 42.7.2: Supported audio protocols*.

If data are received while the preceding received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

Note: The external master components must have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

42.7.7 I2S status flags

Three status flags are provided for the application to fully monitor the state of the I2S bus.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). It indicates the state of the communication layer of the I2S.

When BSY is set, it indicates that the I2S is busy communicating. There is one exception in master receive mode (I2SCFG = 11) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software needs to disable the I2S. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is set when a transfer starts, except when the I2S is in master receiver mode.

The BSY flag is cleared:

- When a transfer completes (except in master transmit mode, in which the communication is supposed to be continuous)
- When the I2S is disabled

When communication is continuous:

- In master transmit mode, the BSY flag is kept high during all the transfers
- In slave mode, the BSY flag goes low for one I2S clock cycle between each transfer

Note: Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

Tx buffer empty flag (TXE)

When set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can then be loaded into it. The TXE flag is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I2S is disabled (I2SE bit is reset).

RX buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when SPIx_DR register is read.

Channel Side flag (CHSIDE)

In transmission mode, this flag is refreshed when TXE goes high. It indicates the channel side to which the data to transfer on SD has to belong. In case of an underrun error event in slave transmission mode, this flag is not reliable and I2S needs to be switched off and switched on before resuming the communication.

In reception mode, this flag is refreshed when data are received into SPIx_DR. It indicates from which channel side data have been received. Note that in case of error (like OVR) this flag becomes meaningless and the I2S must be reset by disabling and then enabling it (with configuration if it needs changing).

This flag has no meaning in the PCM standard (for both Short and Long frame modes).

When the OVR or UDR flag in the SPIx_SR is set and the ERRIE bit in SPIx_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPIx_SR status register (once the interrupt source has been cleared).

42.7.8 I2S error flags

There are three error flags for the I2S cell.

Underrun flag (UDR)

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPIx_DR. It is available when the I2SMOD bit in the SPIx_I2SCFGR register is set. An interrupt may be generated if the ERRIE bit in the SPIx_CR2 register is set.
The UDR bit is cleared by a read operation on the SPIx_SR register.

Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from the SPIx_DR register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx_CR2 register.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPIx_DR register followed by a read access to the SPIx_SR register.

Frame error flag (FRE)

This flag can be set by hardware only if the I2S is configured in Slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the synchronization is lost, the following steps are required to recover from this state and resynchronize the external master device with the I2S slave device:

1. Disable the I2S.
2. Enable it again when the correct level is detected on the WS line (WS line is high in I^2S mode or low for MSB- or LSB-justified or PCM modes).

Desynchronization between master and slave devices may be due to noisy environment on the CK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by software when the status register is read.

42.7.9 DMA features

In I^2S mode, the DMA works in exactly the same way as it does in SPI mode. There is no difference except that the CRC feature is not available in I^2S mode since there is no data transfer protection system.

42.8 I2S interrupts

Table 472 provides the list of I2S interrupts.

Table 472. I2S interrupt requests

Interrupt event	Event flag	Enable control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Overrun error	OVR	ERRIE
Underrun error	UDR	
Frame error flag	FRE	

42.9 SPI and I2S registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI_DR in addition can be accessed by 8-bit access.

42.9.1 SPI and I2S memory map

Table 473. SPI and I2S register memory map

Offset	Register name
0x00	*SPI control register 1 (SPIx_CR1)*
0x04	*SPI control register 2 (SPIx_CR2)*
0x08	*SPI status register (SPIx_SR)*
0x0C	*SPI data register (SPIx_DR)*
0x10	*SPI CRC polynomial register (SPIx_CRCPR)*
0x14	*SPI Rx CRC register (SPIx_RXCRCR)*
0x18	*SPI Tx CRC register (SPIx_TXCRCR)*
0x1C	*SPIx_I2S configuration register (SPIx_I2SCFGR)*
0x20	*SPIx_I2S prescaler register (SPIx_I2SPR)*

42.9.2 SPI control register 1 (SPIx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE:** Bidirectional data mode enable.
This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.
0: 2-line unidirectional data mode selected
1: 1-line bidirectional data mode selected
Note: This bit is not used in I^2S mode.

Bit 14 **BIDIOE:** Output enable in bidirectional mode
This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode.
0: Output disabled (receive-only mode)
1: Output enabled (transmit-only mode)
Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used. This bit is not used in I^2S mode.

Bit 13 **CRCEN:** Hardware CRC calculation enable

0: CRC calculation disabled

1: CRC calculation enabled

Note: This bit must be written only when SPI is disabled (SPE = '0') for correct operation.

This bit is not used in I^2S mode.

Bit 12 **CRCNEXT:** Transmit CRC next

0: Next transmit value is from Tx buffer.

1: Next transmit value is from Tx CRC register.

Note: This bit has to be written as soon as the last data is written in the SPIx_DR register.

This bit is not used in I^2S mode.

Bit 11 **CRCL:** CRC length

This bit is set and cleared by software to select the CRC length.

0: 8-bit CRC length

1: 16-bit CRC length

Note: This bit must be written only when SPI is disabled (SPE = '0') for correct operation.

This bit is not used in I^2S mode.

Bit 10 **RXONLY:** Receive only mode enabled.

This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receive only mode is active.This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

0: Full-duplex (Transmit and receive)

1: Output disabled (Receive-only mode)

Note: This bit is not used in I^2S mode.

Bit 9 **SSM:** Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

0: Software slave management disabled

1: Software slave management enabled

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 8 **SSI:** Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored.

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 7 **LSBFIRST***:* Frame format

0: data is transmitted / received with the MSB first

1: data is transmitted / received with the LSB first

Note: 1. This bit must not be changed when communication is ongoing.

2. This bit is not used in I^2S mode and SPI TI mode.

Bit 6 **SPE:** SPI enable

0: Peripheral disabled

1: Peripheral enabled

Note: When disabling the SPI, follow the procedure described in Procedure for disabling the SPI.

This bit is not used in I^2S mode.

Bits 5:3 **BR[2:0]:** Baud rate control
000: $f_{PCLK}/2$
001: $f_{PCLK}/4$
010: $f_{PCLK}/8$
011: $f_{PCLK}/16$
100: $f_{PCLK}/32$
101: $f_{PCLK}/64$
110: $f_{PCLK}/128$
111: $f_{PCLK}/256$
Note: These bits must not be changed when communication is ongoing.
These bits are not used in I^2S mode.

Bit 2 **MSTR:** Master selection
0: Slave configuration
1: Master configuration
Note: This bit must not be changed when communication is ongoing.
This bit is not used in I^2S mode.

Bit 1 **CPOL:** Clock polarity
0: CK to 0 when idle
1: CK to 1 when idle
Note: This bit must not be changed when communication is ongoing.
This bit is not used in I^2S mode and SPI TI mode except the case when CRC is applied at TI mode.

Bit 0 **CPHA:** Clock phase
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge
Note: This bit must not be changed when communication is ongoing.
This bit is not used in I^2S mode and SPI TI mode except the case when CRC is applied at TI mode.

42.9.3 SPI control register 2 (SPIx_CR2)

Address offset: 0x04

Reset value: 0x0700

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **LDMA_TX:** Last DMA transfer for transmission

This bit is used in data packing mode, to define if the total number of data to transmit by DMA is odd or even. It has significance only if the TXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).

0: Number of data to transfer is even

1: Number of data to transfer is odd

Note: Refer to Procedure for disabling the SPI if the CRCEN bit is set.

This bit is not used in I²S mode.

Bit 13 **LDMA_RX**: Last DMA transfer for reception

This bit is used in data packing mode, to define if the total number of data to receive by DMA is odd or even. It has significance only if the RXDMAEN bit in the SPIx_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx_CR1 register).

0: Number of data to transfer is even

1: Number of data to transfer is odd

Note: Refer to Procedure for disabling the SPI if the CRCEN bit is set.

This bit is not used in I²S mode.

Bit 12 **FRXTH**: FIFO reception threshold

This bit is used to set the threshold of the RXFIFO that triggers an RXNE event

0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit)

1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)

Note: This bit is not used in I^2S mode

Bits 11:8 **DS[3:0]**: Data size

These bits configure the data length for SPI transfers.

0000: Not used

0001: Not used

0010: Not used

0011: 4-bit

0100: 5-bit

0101: 6-bit

0110: 7-bit

0111: 8-bit

1000: 9-bit

1001: 10-bit

1010: 11-bit

1011: 12-bit

1100: 13-bit

1101: 14-bit

1110: 15-bit

1111: 16-bit

If software attempts to write one of the “Not used” values, they are forced to the value “0111” (8-bit)

Note: These bits are not used in I^2S mode.

Bit 7 **TXEIE:** Tx buffer empty interrupt enable

0: TXE interrupt masked

1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE:** RX buffer not empty interrupt enable

0: RXNE interrupt masked

1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE:** Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode and UDR, OVR, and FRE in I^2S mode).

0: Error interrupt is masked

1: Error interrupt is enabled

Bit 4 **FRF**: Frame format

0: SPI Motorola mode

1 SPI TI mode

Note: This bit must be written only when the SPI is disabled (SPE=0).
This bit is not used in I^2S mode.

Bit 3 **NSSP**: NSS pulse management

This bit is used in master mode only. It allows the SPI to generate an NSS pulse between two consecutive data when doing continuous transfers. In the case of a single data transfer, it forces the NSS pin high level after the transfer.

It has no meaning if CPHA = ’1’, or FRF = ’1’.

0: No NSS pulse

1: NSS pulse generated

Note: 1. This bit must be written only when the SPI is disabled (SPE=0).
2. This bit is not used in I^2S mode and SPI TI mode.

Bit 2 **SSOE:** SS output enable

0: SS output is disabled in master mode and the SPI interface can work in multimaster configuration

1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multimaster environment.

Note: This bit is not used in I^2S mode and SPI TI mode.

Bit 1 **TXDMAEN:** Tx buffer DMA enable

When this bit is set, a DMA request is generated whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

Bit 0 **RXDMAEN:** Rx buffer DMA enable

When this bit is set, a DMA request is generated whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

42.9.4 SPI status register (SPIx_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
			r	r	r	r	r	r	r	r	rc_w0	r	r	r	r

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:11 **FTLVL[1:0]:** FIFO transmission level
These bits are set and cleared by hardware.
00: FIFO empty
01: 1/4 FIFO
10: 1/2 FIFO
11: FIFO full (considered as FULL when the FIFO threshold is greater than 1/2)
Note: This bit is not used in I^2S mode.

Bits 10:9 **FRLVL[1:0]**: FIFO reception level
These bits are set and cleared by hardware.
00: FIFO empty
01: 1/4 FIFO
10: 1/2 FIFO
11: FIFO full
Note: These bits are not used in I²S mode and in SPI receive-only mode while CRC calculation is enabled.

Bit 8 **FRE**: Frame format error
This flag is used for SPI in TI slave mode and I^2S slave mode. Refer to *Section 42.5.11: SPI error flags* and *Section 42.7.8: I2S error flags*.
This flag is set by hardware and reset when SPIx_SR is read by software.
0: No frame format error
1: A frame format error occurred

Bit 7 **BSY:** Busy flag
0: SPI (or I2S) not busy
1: SPI (or I2S) is busy in communication or Tx buffer is not empty
This flag is set and cleared by hardware.
Note: The BSY flag must be used with caution: refer to Section 42.5.10: SPI status flags and Procedure for disabling the SPI.

Bit 6 **OVR:** Overrun flag
0: No overrun occurred
1: Overrun occurred
This flag is set by hardware and reset by a software sequence. Refer to *I2S error flags* for the software sequence.

Bit 5 **MODF:** Mode fault
0: No mode fault occurred
1: Mode fault occurred
This flag is set by hardware and reset by a software sequence. Refer to *Mode fault (MODF)* for the software sequence.
Note: This bit is not used in I^2S mode.

Bit 4 **CRCERR:** CRC error flag
0: CRC value received matches the SPIx_RXCRCR value
1: CRC value received does not match the SPIx_RXCRCR value
Note: This flag is set by hardware and cleared by software writing 0.
This bit is not used in I^2S mode.

Bit 3 **UDR:** Underrun flag
0: No underrun occurred
1: Underrun occurred
This flag is set by hardware and reset by a software sequence. Refer to *Section 42.7.8: I2S error flags* for the software sequence.
Note: This bit is not used in SPI mode.

Bit 2 **CHSIDE**: Channel side
0: Channel Left has to be transmitted or has been received
1: Channel Right has to be transmitted or has been received
Note: This bit is not used in SPI mode. It has no significance in PCM mode.

Bit 1 **TXE:** Transmit buffer empty
0: Tx buffer not empty
1: Tx buffer empty

Bit 0 **RXNE:** Receive buffer not empty
0: Rx buffer empty
1: Rx buffer not empty

42.9.5 SPI data register (SPIx_DR)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DR[15:0]:** Data register
Data received or to be transmitted
The data register serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses TxFIFO (See *Section 42.5.9: Data transmission and reception procedures*).
Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.

42.9.6 SPI CRC polynomial register (SPIx_CRCPR)

Address offset: 0x10

Reset value: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CRCPOLY[15:0]:** CRC polynomial register
This register contains the polynomial for the CRC calculation.
The CRC polynomial (0x0007) is the reset value of this register. Another polynomial can be configured as required.

Note: The polynomial value must be odd only. No even value is supported.

42.9.7 SPI Rx CRC register (SPIx_RXCRCR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **RXCRC[15:0]:** Rx CRC register

When CRC calculation is enabled, the RXCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPIx_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16 bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY Flag is set can return an incorrect value.

These bits are not used in I^2S mode.

42.9.8 SPI Tx CRC register (SPIx_TXCRCR)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **TXCRC[15:0]:** Tx CRC register

When CRC calculation is enabled, the TXCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPIx_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16 bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY flag is set can return an incorrect value.

These bits are not used in I^2S mode.

42.9.9 SPIx_I2S configuration register (SPIx_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ASTRTEN	I2SMOD	I2SE	I2SCFG[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		CKPOL	DATLEN[1:0]		CHLEN
			rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ASTRTEN**: Asynchronous start enable.

0: The Asynchronous start is disabled.

When the I2S is enabled in slave mode, the hardware starts the transfer when the I2S clock is received and an appropriate transition is detected on the WS signal.

1: The Asynchronous start is enabled.

When the I2S is enabled in slave mode, the hardware starts the transfer when the I2S clock is received and the appropriate level is detected on the WS signal.

*Note: The appropriate **transition** is a falling edge on WS signal when I^2S Philips Standard is used, or a rising edge for other standards.*

*The appropriate **level** is a low level on WS signal when I^2S Philips Standard is used, or a high level for other standards.*

Please refer to Section 42.7.3: Startup description for additional information.

Bit 11 **I2SMOD**: I2S mode selection

0: SPI mode is selected

1: I2S mode is selected

Note: This bit must be configured when the SPI is disabled.

Bit 10 **I2SE**: I2S enable

0: I2S peripheral is disabled

1: I2S peripheral is enabled

Note: This bit is not used in SPI mode.

Bits 9:8 **I2SCFG[1:0]**: I2S configuration mode

00: Slave - transmit

01: Slave - receive

10: Master - transmit

11: Master - receive

Note: These bits must be configured when the I2S is disabled.

They are not used in SPI mode.

Bit 7 **PCMSYNC**: PCM frame synchronization

0: Short frame synchronization

1: Long frame synchronization

Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used).

It is not used in SPI mode.

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I2S standard selection
00: I^{2}S Philips standard
01: MSB justified standard (left justified)
10: LSB justified standard (right justified)
11: PCM standard
For more details on I^{2}S standards, refer to *Section 42.7.2: Supported audio protocols*
Note: For correct operation, these bits must be configured when the I2S is disabled.
They are not used in SPI mode.

Bit 3 **CKPOL**: Inactive state clock polarity
0: I2S clock inactive state is low level
1: I2S clock inactive state is high level
Note: For correct operation, this bit must be configured when the I2S is disabled.
It is not used in SPI mode.
The bit CKPOL does not affect the CK edge sensitivity used to receive or transmit the SD and WS signals.

Bits 2:1 **DATLEN[1:0]**: Data length to be transferred
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: For correct operation, these bits must be configured when the I2S is disabled.
They are not used in SPI mode.

Bit 0 **CHLEN**: Channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide
The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in.
Note: For correct operation, this bit must be configured when the I2S is disabled.
It is not used in SPI mode.

42.9.10 SPIx_I2S prescaler register (SPIx_I2SPR)

Address offset: 0x20

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **MCKOE**: Master clock output enable
0: Master clock output is disabled
1: Master clock output is enabled
Note: This bit must be configured when the I2S is disabled. It is used only when the I2S is in master mode.
It is not used in SPI mode.

Bit 8 **ODD**: Odd factor for the prescaler

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2) + 1

Refer to *Section 42.7.3: Startup description*.

Note: This bit must be configured when the I2S is disabled. It is used only when the I2S is in master mode.

It is not used in SPI mode.

Bits 7:0 **I2SDIV[7:0]**: I2S linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to *Section 42.7.3: Startup description*.

Note: These bits must be configured when the I2S is disabled. They are used only when the I2S is in master mode.

They are not used in SPI mode.

43 Inter-integrated circuit (I2C) interface

43.1 Introduction

The I^2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I^2C bus. It provides multimaster capability, and controls all I^2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), and Fast-mode (Fm).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

43.2 I2C main features

- I^2C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available depending on the product implementation (see *Section 43.3: I2C implementation*):

- SMBus specification rev 3.0 compatibility:
  - Hardware PEC (Packet Error Checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and Device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming

43.3 I2C implementation

This manual describes the full set of features implemented in I2C peripheral. In the SR5E1x devices I2C1 and I2C2 implement the full set of features as shown in the following table.

Table 474. I2C implementation

I2C features(1)	I2C1	I2C2
7-bit addressing mode	X	X
10-bit addressing mode	X	X
Standard-mode (up to 100 kbit/s)	X	X
Fast-mode (up to 400 kbit/s)	X	X
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	No	No
Independent clock	X	X
SMBus/PMBus	X	X

1. X = supported.

43.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz) or Fast-mode (up to 400 kHz) I^2C bus.

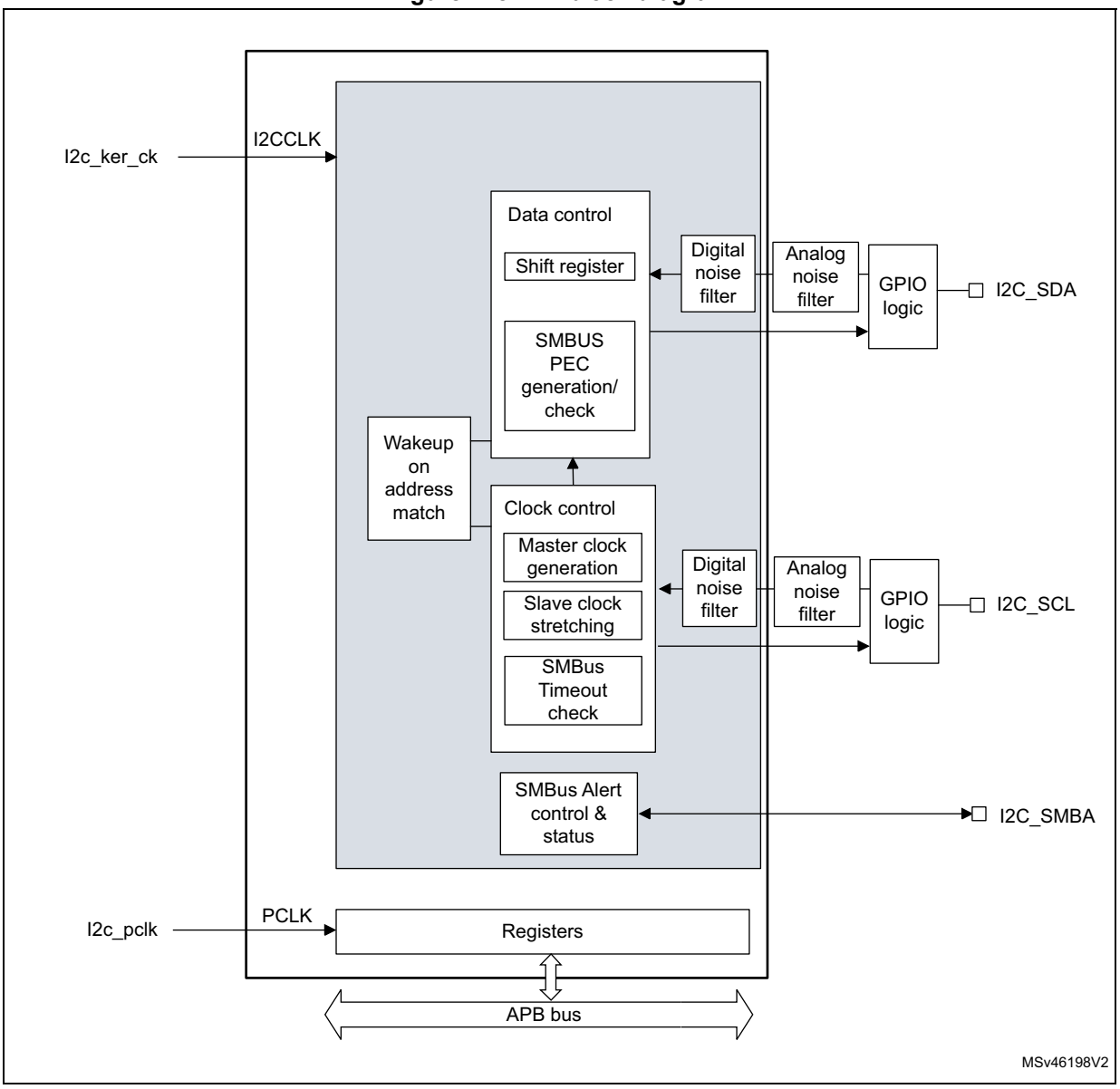
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

43.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 718*.

Figure 718. I2C block diagram



The I2C is clocked by an independent clock source which allows the I2C to operate independently from the PCLK frequency.

43.4.2 I2C pins and internal signals

Table 475. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I2C data
I2C_SCL	Bidirectional	I2C clock
I2C_SMBA	Bidirectional	SMBus Alert

Table 476. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to *Table 490: I2C Interrupt requests* for the full list of interrupt sources
i2c_rx_dma	Output	I2C Receive Data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C Transmit Data DMA request (I2C_TX)

43.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$ and $t_{I2CCLK} < t_{HIGH}$

with:

t_{LOW}: SCL low time and t_{HIGH}: SCL high time

$t_{filters}$: when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is DNF x t_{I2CCLK}.

The PCLK clock period t_{PCLK} must respect the following condition:

$t_{PCLK} < 4/3\ t_{SCL}$

with t_{SCL}: SCL period

Caution: When the I2C kernel is clocked by PCLK, this clock must respect the conditions for t_{I2CCLK}.

43.4.4 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

Communication flow

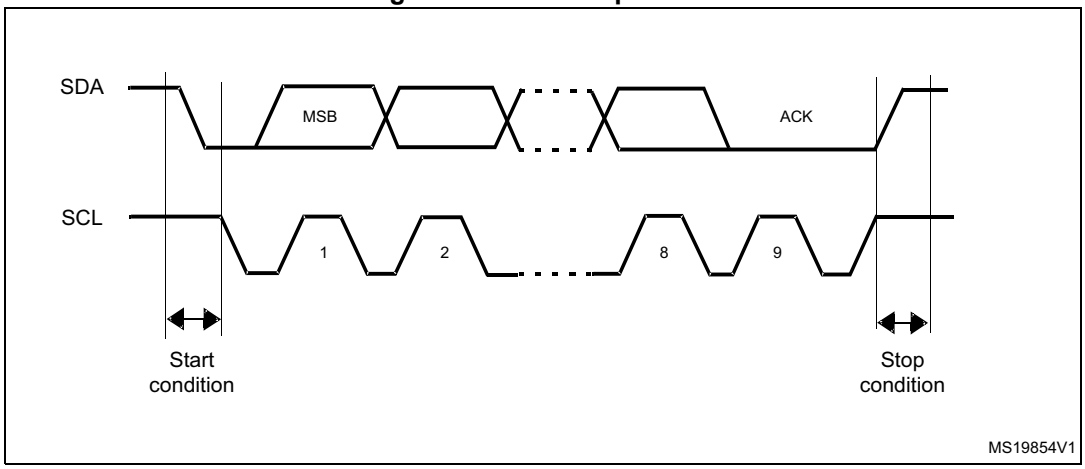
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Figure 719. I^2C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

43.4.5 I2C initialization

Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller.

Then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE=0), the I^2C performs a software reset. Refer to *Section 43.4.6: Software reset* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I^2C specification which requires the

suppression of spikes with a pulse width up to 50 ns in Fast-mode. The user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows spikes with a programmable length of 1 to 15 I2CCLK periods to be suppressed.

Table 477. Comparison of analog vs. digital filters

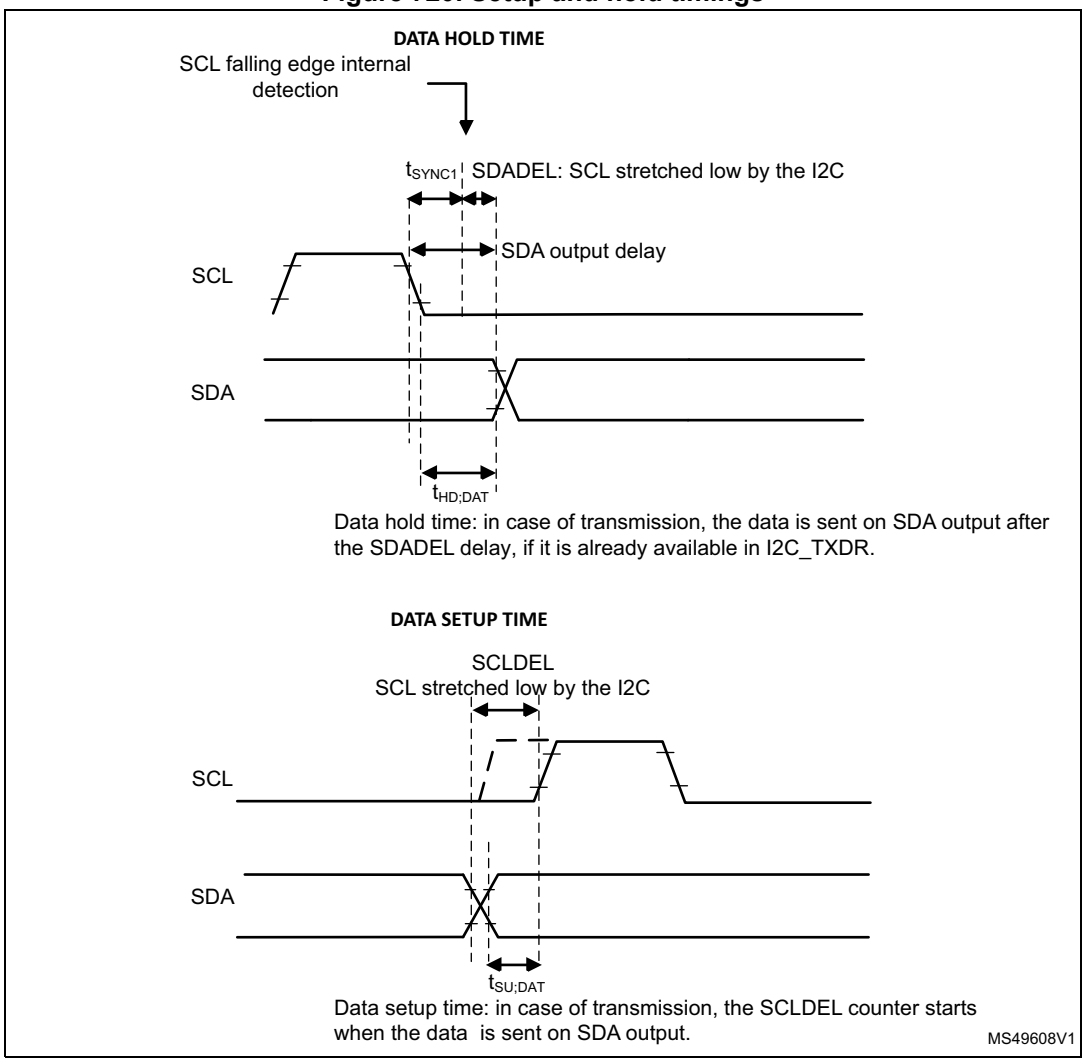
-	Analog filter	Digital filter
Pulse width of suppressed spikes	≥ 50 ns	Programmable length from 1 to 15 I2C peripheral clocks
Benefits	Available in Stop mode	– Programmable length: extra filtering capability vs. standard requirements – Stable length
Drawbacks	Variation vs. temperature, voltage, process	Wakeup from Stop mode on address match is not available when digital filter is enabled

Caution: Changing the filter configuration is not allowed when the I2C is enabled.

I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

Figure 720. Setup and hold timings



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  T_{SDADEL} impacts the hold time $t_{HD;DAT}$.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC+1) + 1] x t_{I2CCLK} }

t_{SYNC1} duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter: $t_{AF(min)} < t_{AF} < t_{AF(max)}$
- When enabled, input delay brought by the digital filter: t_{DNF} = DNF x t_{I2CCLK}
- Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

$\{t_{f\ (max)} + t_{HD;DAT\ (min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: *$t_{AF(min)}$ / $t_{AF(max)}$ are part of the equation only when the analog filter is enabled. Refer to device datasheet for t_{AF} values.*

The maximum $t_{HD;DAT}$ can be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode, but must be less than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - 260\ ns - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$.

Note: *This condition can be violated when NOSTRETCH=0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.*

Refer to *Table 478: I2C-SMBUS specification data setup and hold times* for t_f, t_r, $t_{HD;DAT}$ and $t_{VD;DAT}$ standard values.

- After t_{SDADEL} delay, or after sending SDA output in case the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, SCL line is kept at low level during the setup time. This setup time is t_{SCLDEL} = (SCLDEL+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC+1)] \times t_{I2CCLK}]\} - 1 <= SCLDEL$

Refer to *Table 478: I2C-SMBUS specification data setup and hold times* for t_r and $t_{SU;DAT}$ standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application is.

Note: At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, in case the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOSTRETCH=1 in slave mode, the SCL is not stretched. Consequently the SDADEL must be programmed in such a way to guarantee also a sufficient setup time.

Table 478. I^2C-SMBUS specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		SMBUS		Unit
		Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	300	

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is t_{SCLL} = (SCLL+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.
  t_{SCLL} impacts the SCL low time t_{LOW}.
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is t_{SCLH} = (SCLH+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}. t_{SCLH} impacts the SCL high time t_{HIGH}.

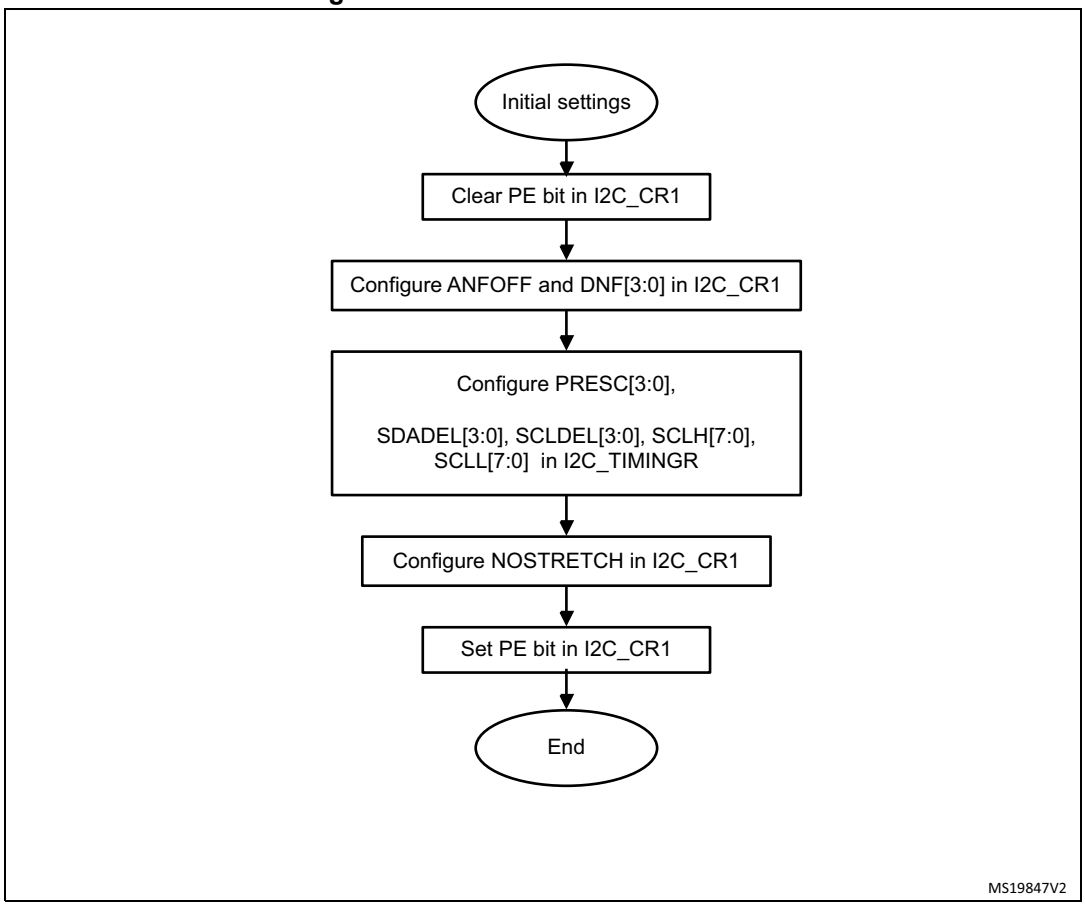
Refer to *I2C master initialization* for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to *I2C slave initialization* for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 721. I2C initialization flowchart



43.4.6 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1.

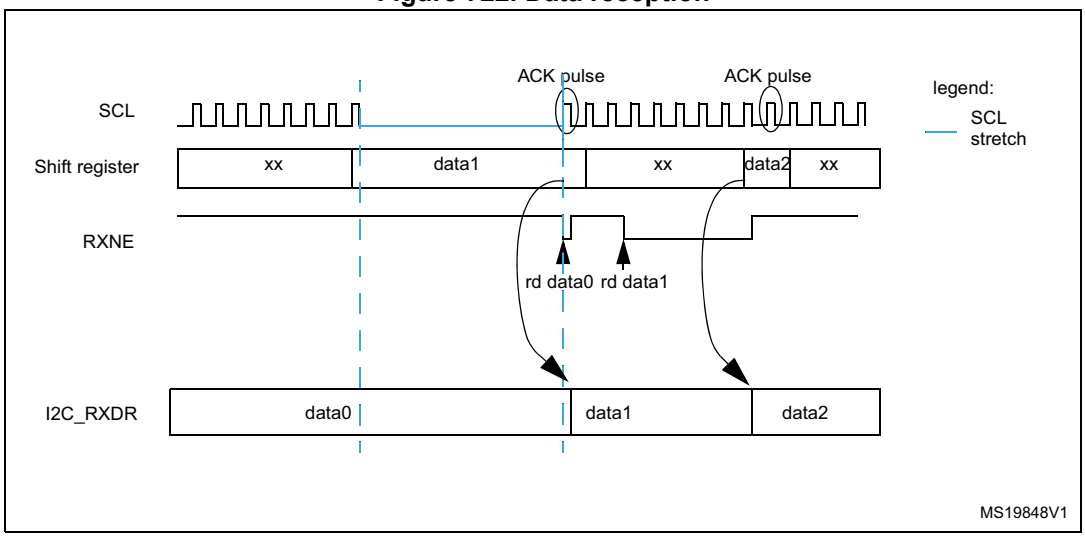
43.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

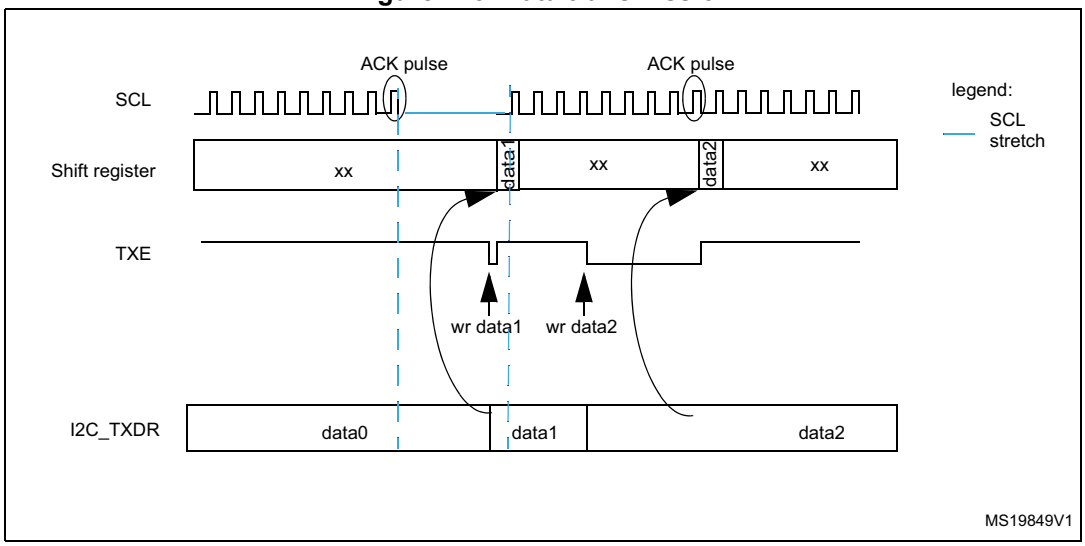
Figure 722. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

Figure 723. Data transmission



Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (Slave Byte Control) bit in the I2C_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in master mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = ‘1’ in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.
- **Software end mode** (AUTOEND = ‘0’ in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the master wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 479. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Master Tx/Rx NBYTES + STOP	x	0	1
Master Tx/Rx + NBYTES + RESTART	x	0	0
Slave Tx/Rx all received bytes ACKed	0	x	x
Slave Rx with ACK control	1	1	x

43.4.8 I2C slave mode

I2C slave initialization

In order to work in slave mode, the user must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.

  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.
- If additional slave addresses are required, the 2nd slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.
- The General Call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE=1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}.

Slave without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Slave Byte Control mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

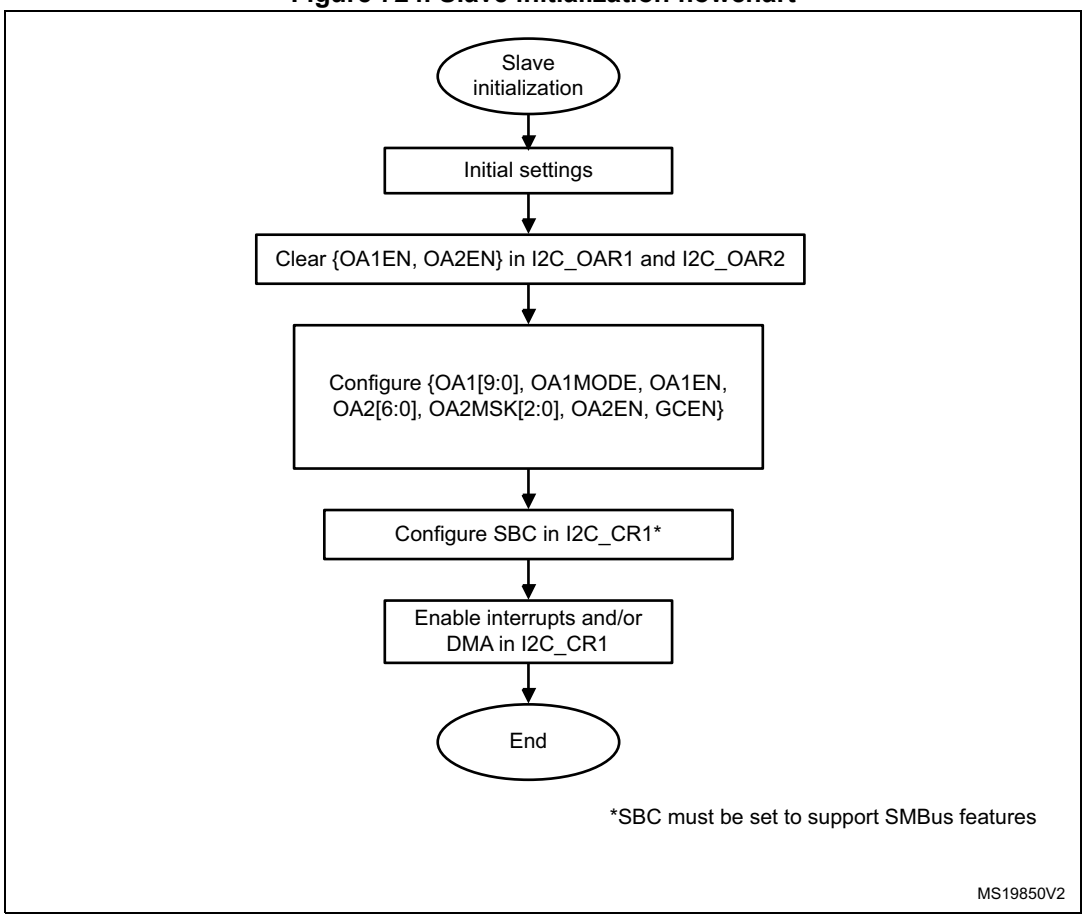
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

Caution: Slave Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

Figure 724. Slave initialization flowchart



Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to ‘0’. In this case, If TXE = 0 when the slave address is received (ADDR=1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave Byte Control mode (SBC=1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR=1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH=1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event is needed, (Transmit Interrupt or Transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

Figure 725. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 0

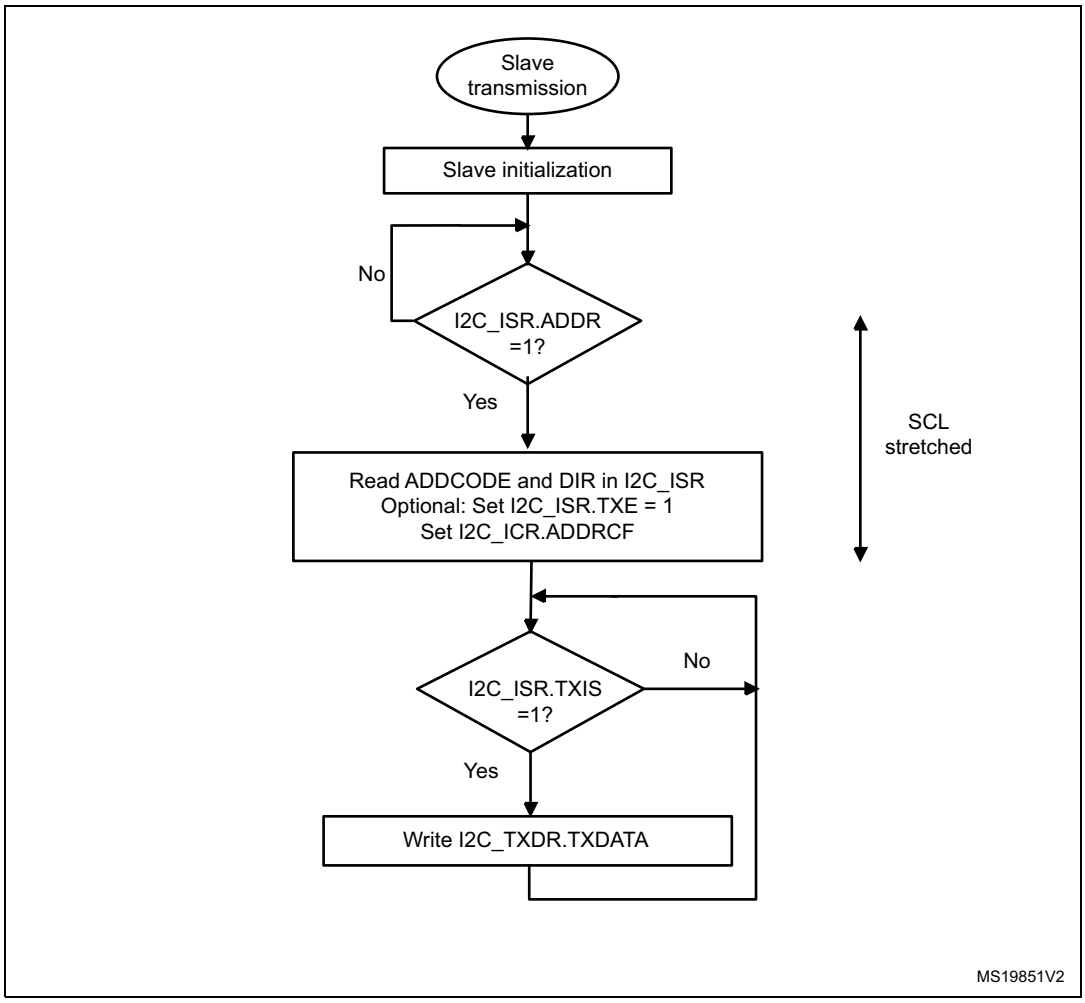


Figure 726. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH= 1

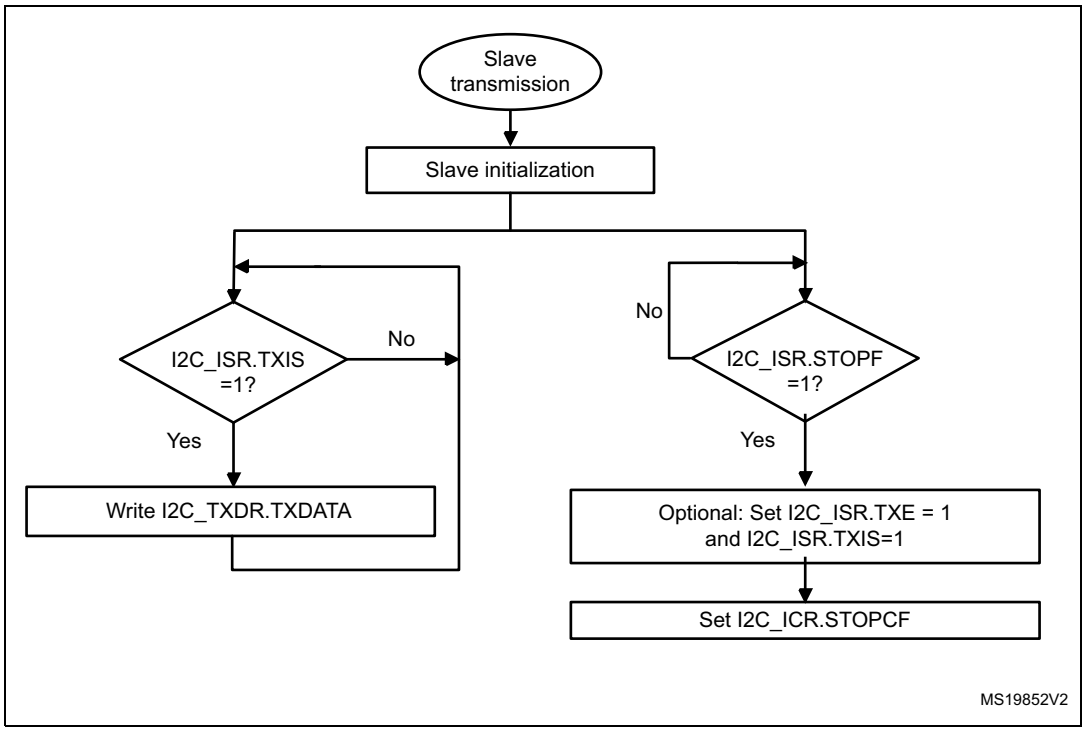
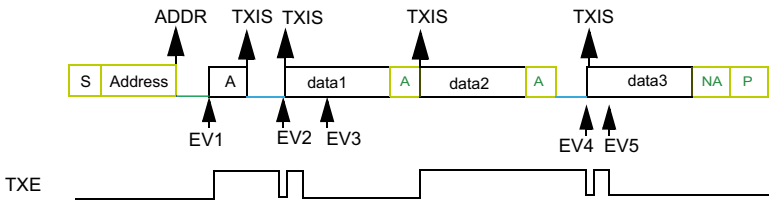


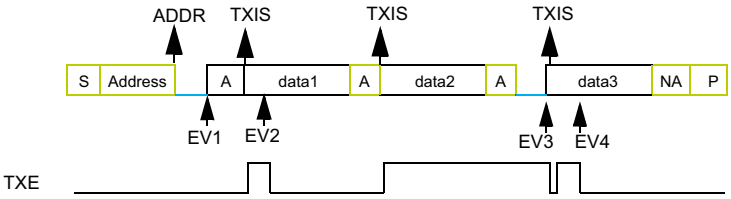
Figure 727. Transfer bus diagrams for I2C slave transmitter

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



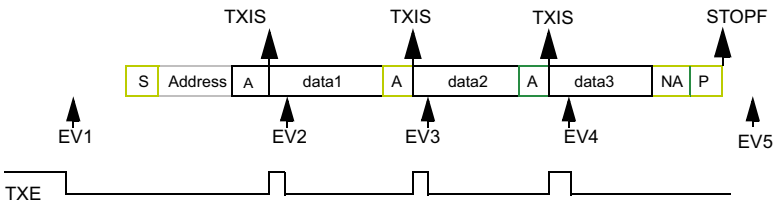
EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:



EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MS19853V2

Slave receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 728. Transfer sequence flowchart for slave receiver with NOSTRETCH=0

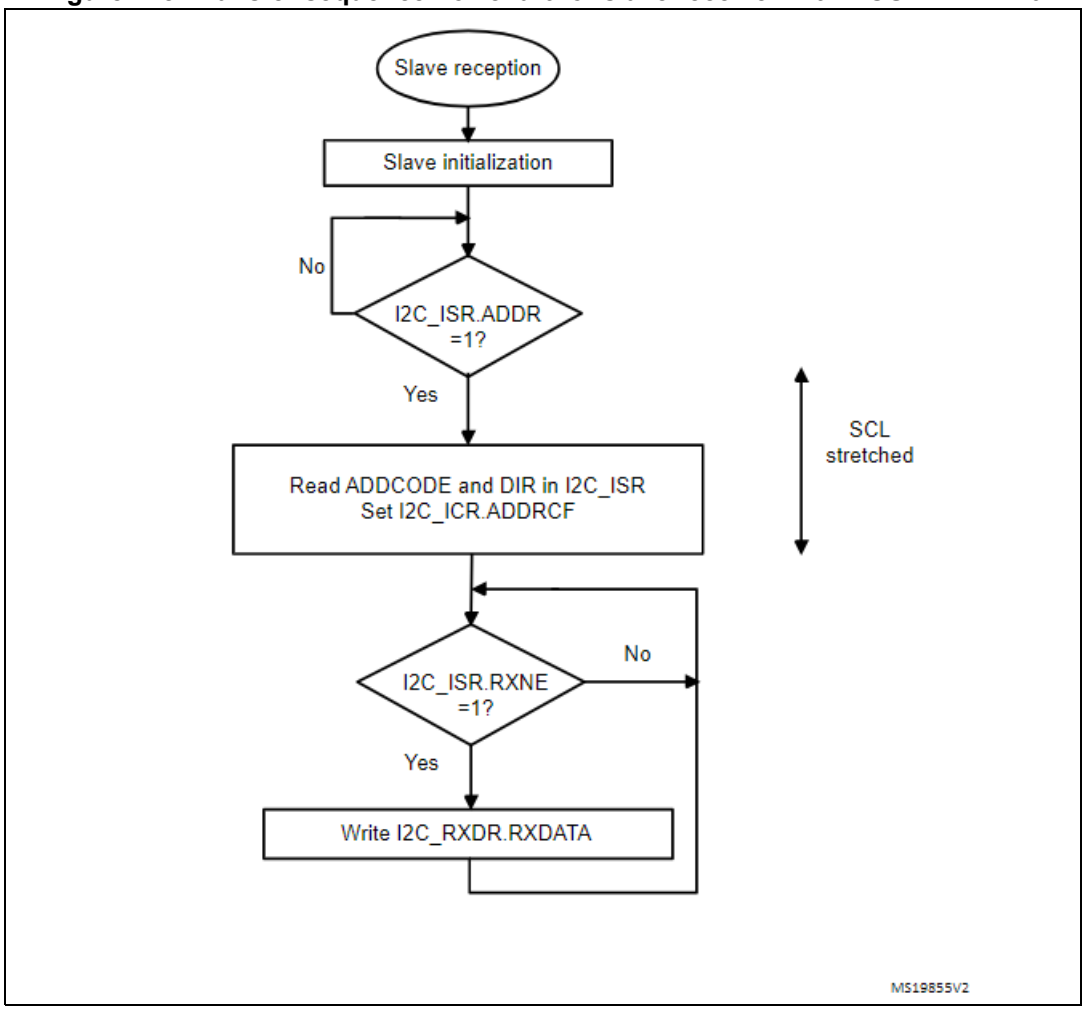


Figure 729. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

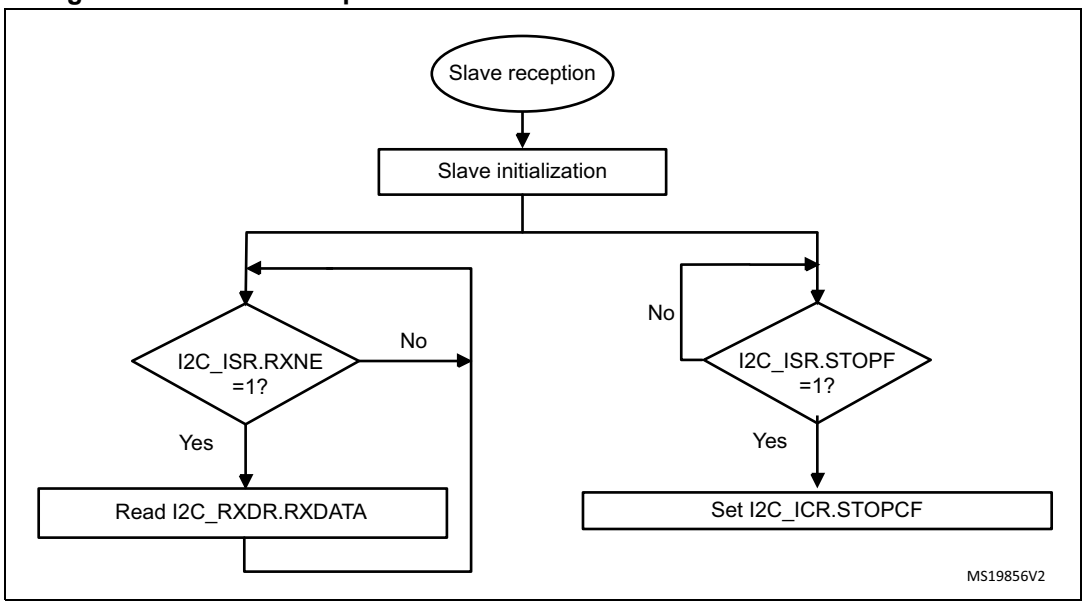
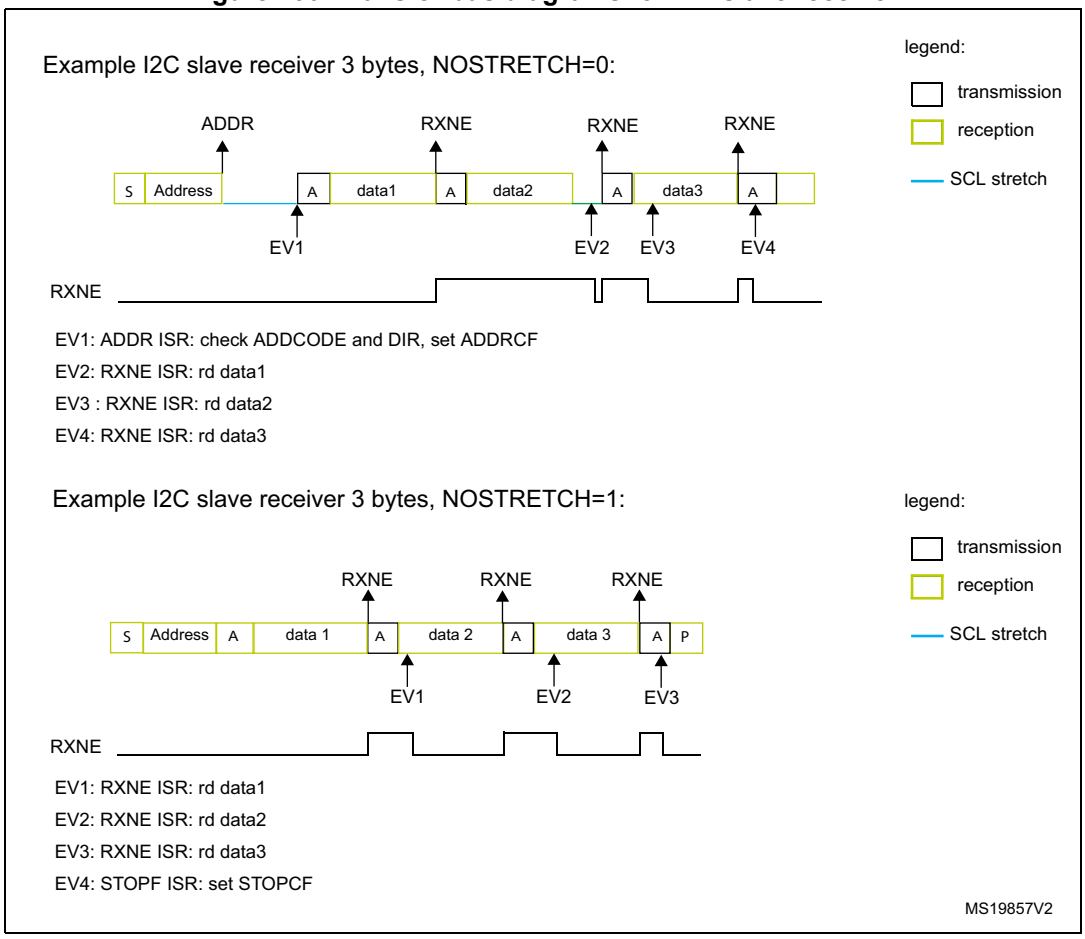


Figure 730. Transfer bus diagrams for I2C slave receiver



43.4.9 I2C master mode

I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the master clock period is:

$$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$$

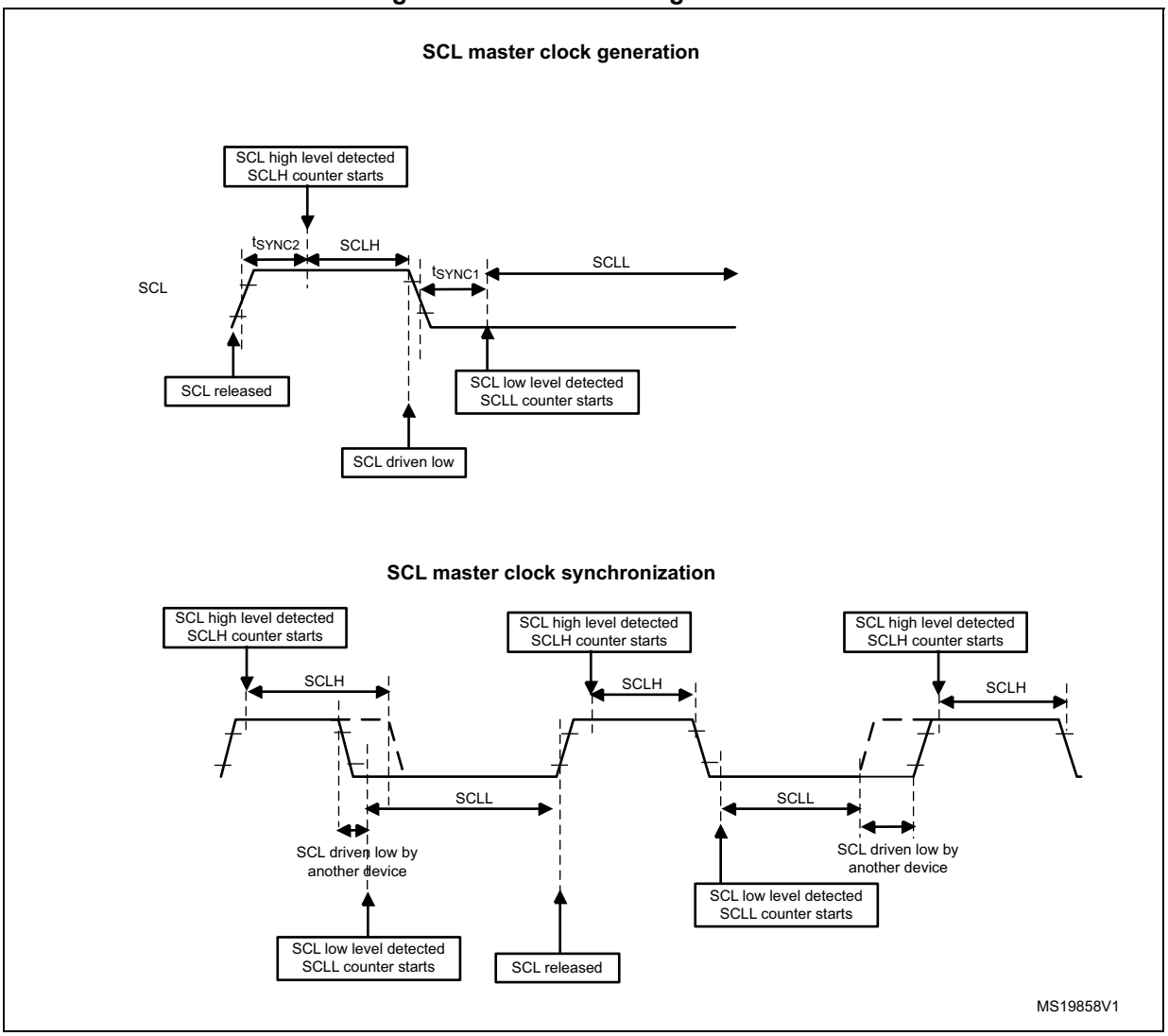
The duration of t_{SYNC1} depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of t_{SYNC2} depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

Figure 731. Master clock generation



Caution: In order to be I^{2}C or SMBus compliant, the master clock must respect the timings given in the table below.

Table 480. I²C-SMBUS specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		SMBUS		Unit
		Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	4.7	-	µs
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	4.0	-	µs
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	4.7	-	µs
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	4.7	-	µs
t_{HIGH}	Period of the SCL clock	4.0	-	0.6	-	4.0	50	µs
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	300	ns

Note: SCLL is also used to generate the t_{BUF} and $t_{SU:STA}$ timings.

SCLH is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 43.4.10: I2C_TIMINGR register configuration examples* for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configured to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

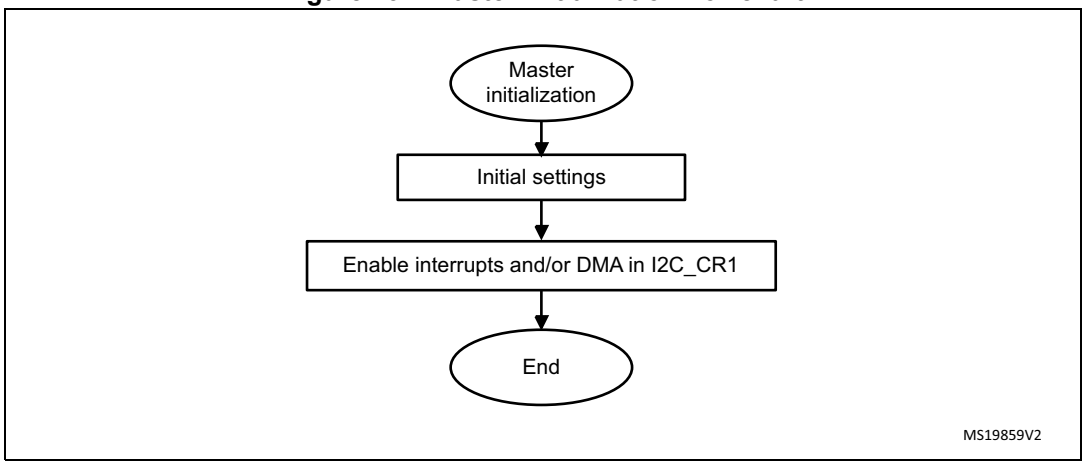
Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t_{BUF}.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

Note: The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value is. The START bit is also reset by hardware if an arbitration loss occurs.
In 10-bit addressing mode, when the Slave Address first 7 bits is NACKed by the slave, the master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, in order to stop sending the slave address.
If the I2C is addressed as a slave (ADDR=1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared, when the ADDRCF bit is set.

Note: The same procedure is applied for a Repeated Start condition. In this case BUSY=1.

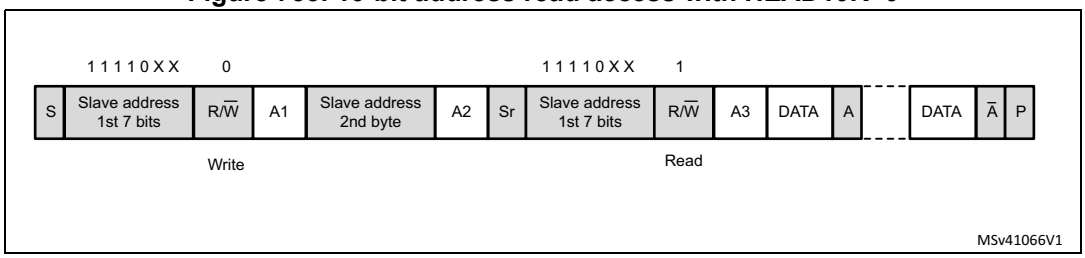
Figure 732. Master initialization flowchart



Initialization of a master receiver addressing a 10-bit address slave

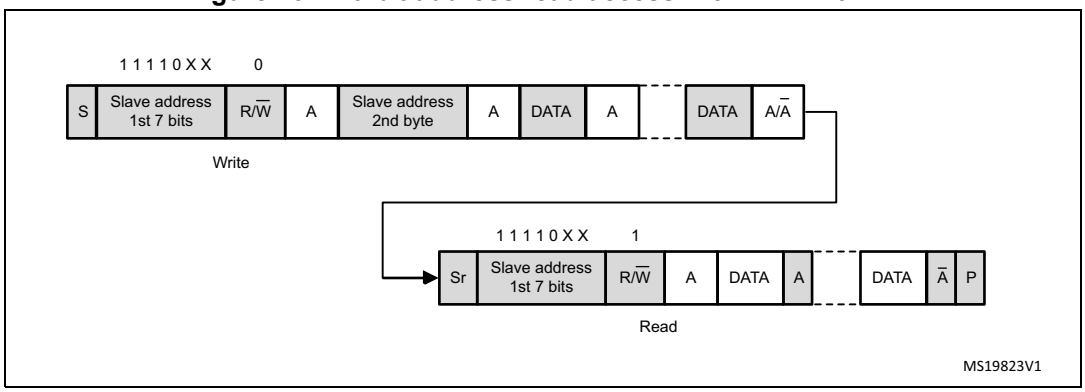
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address 2nd byte + REStart + Slave address 10-bit header Read.

Figure 733. 10-bit address read access with HEAD10R=0



- If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10-bit slave address configured with HEAD10R=1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

Figure 734. 10-bit address read access with HEAD10R=1



Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. The NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.

Figure 735. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes

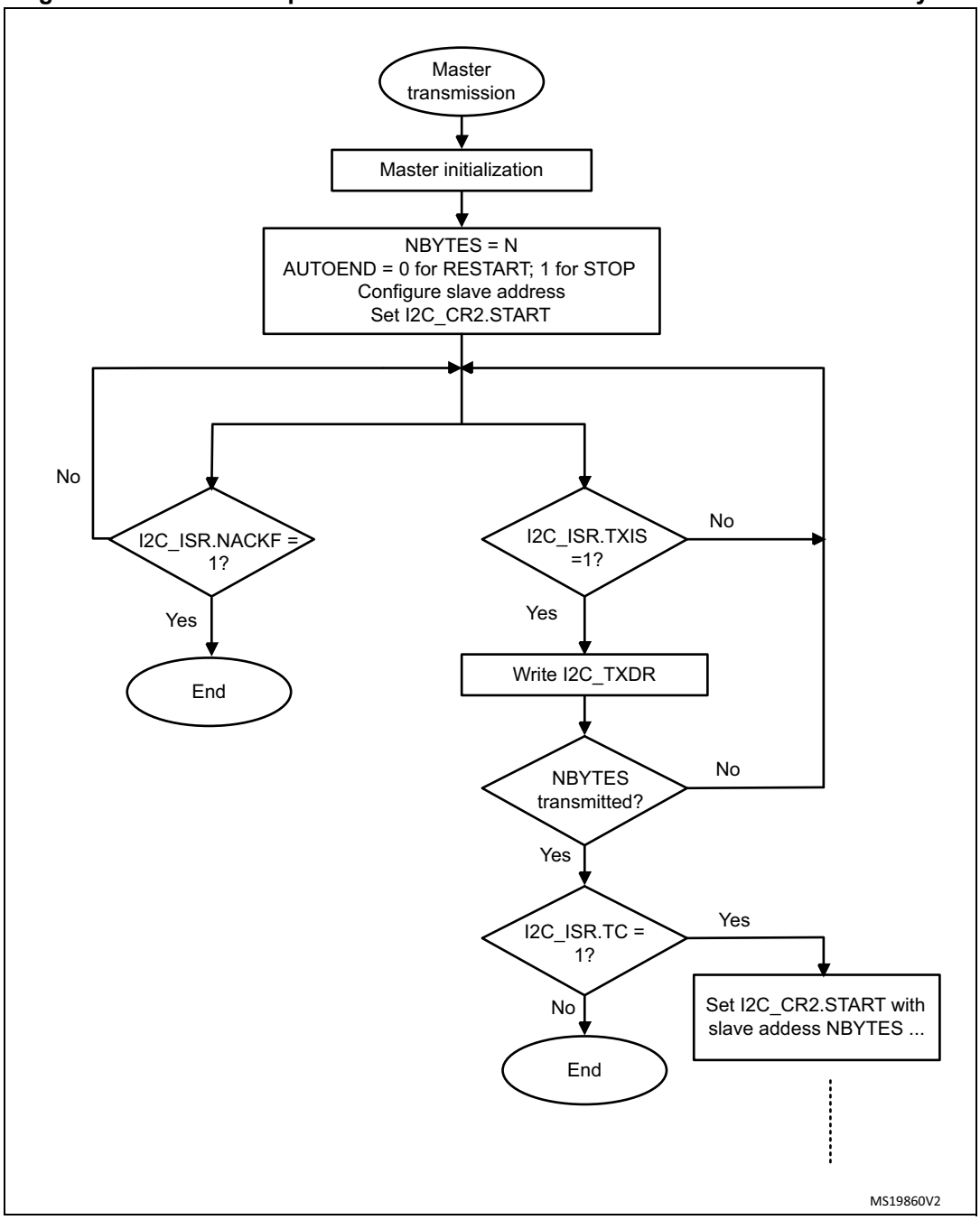


Figure 736. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

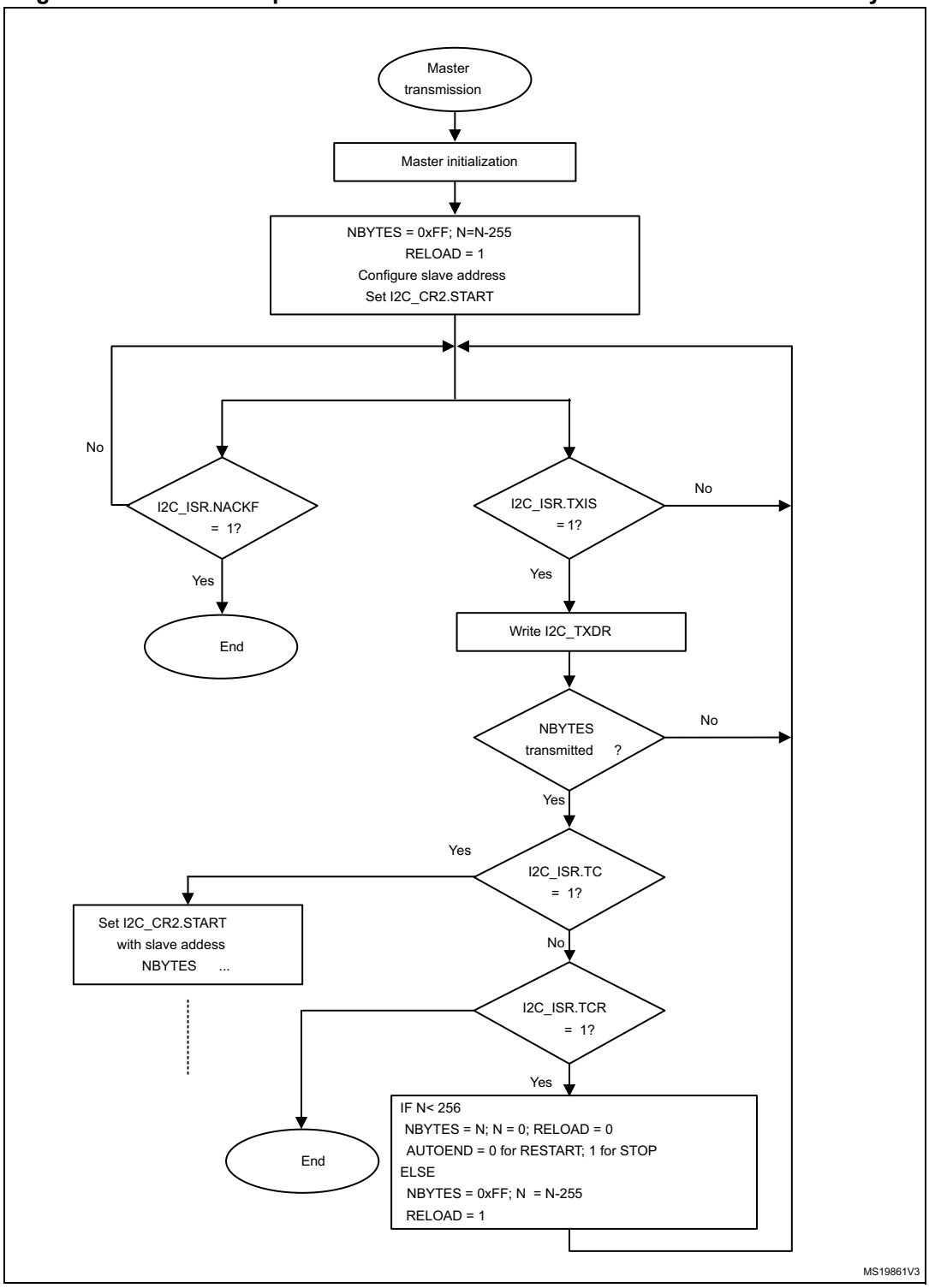
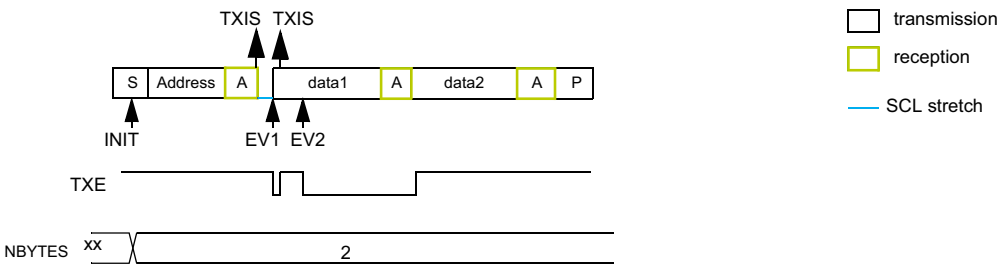


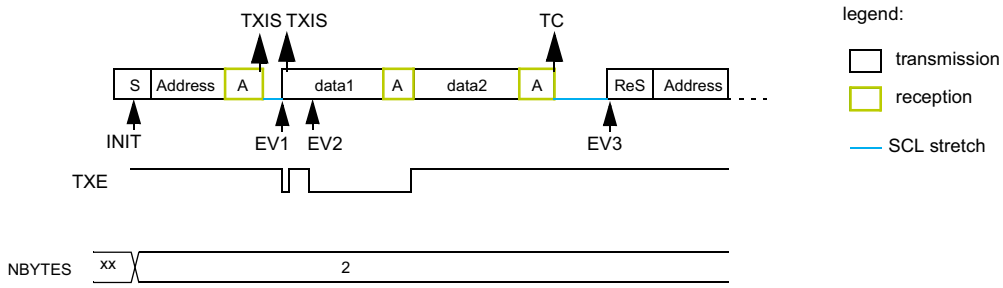
Figure 737. Transfer bus diagrams for I2C master transmitter

Example I2C master transmitter 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19862V2

Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 738. Transfer sequence flowchart for I2C master receiver for N≤255 bytes

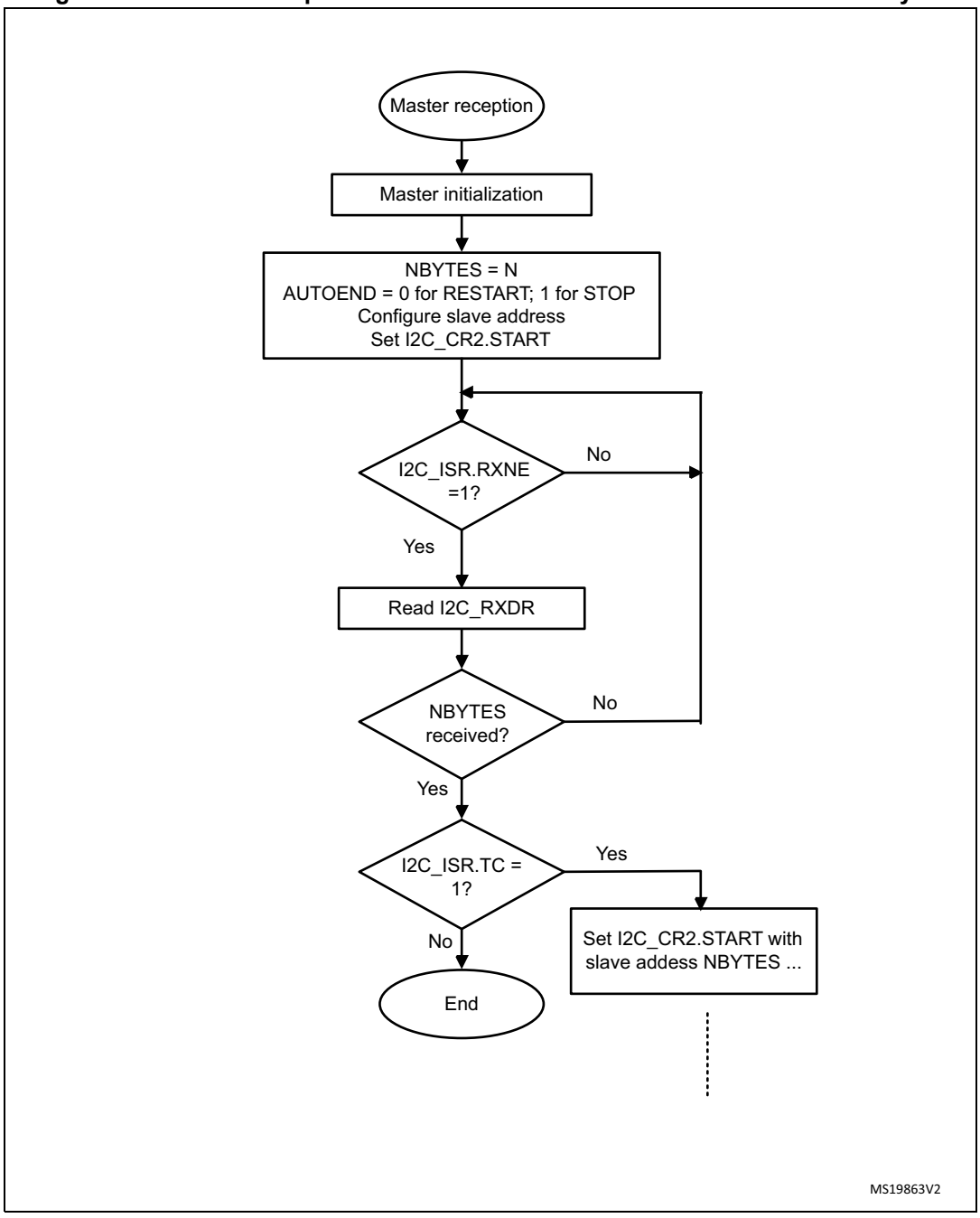


Figure 739. Transfer sequence flowchart for I2C master receiver for N >255 bytes

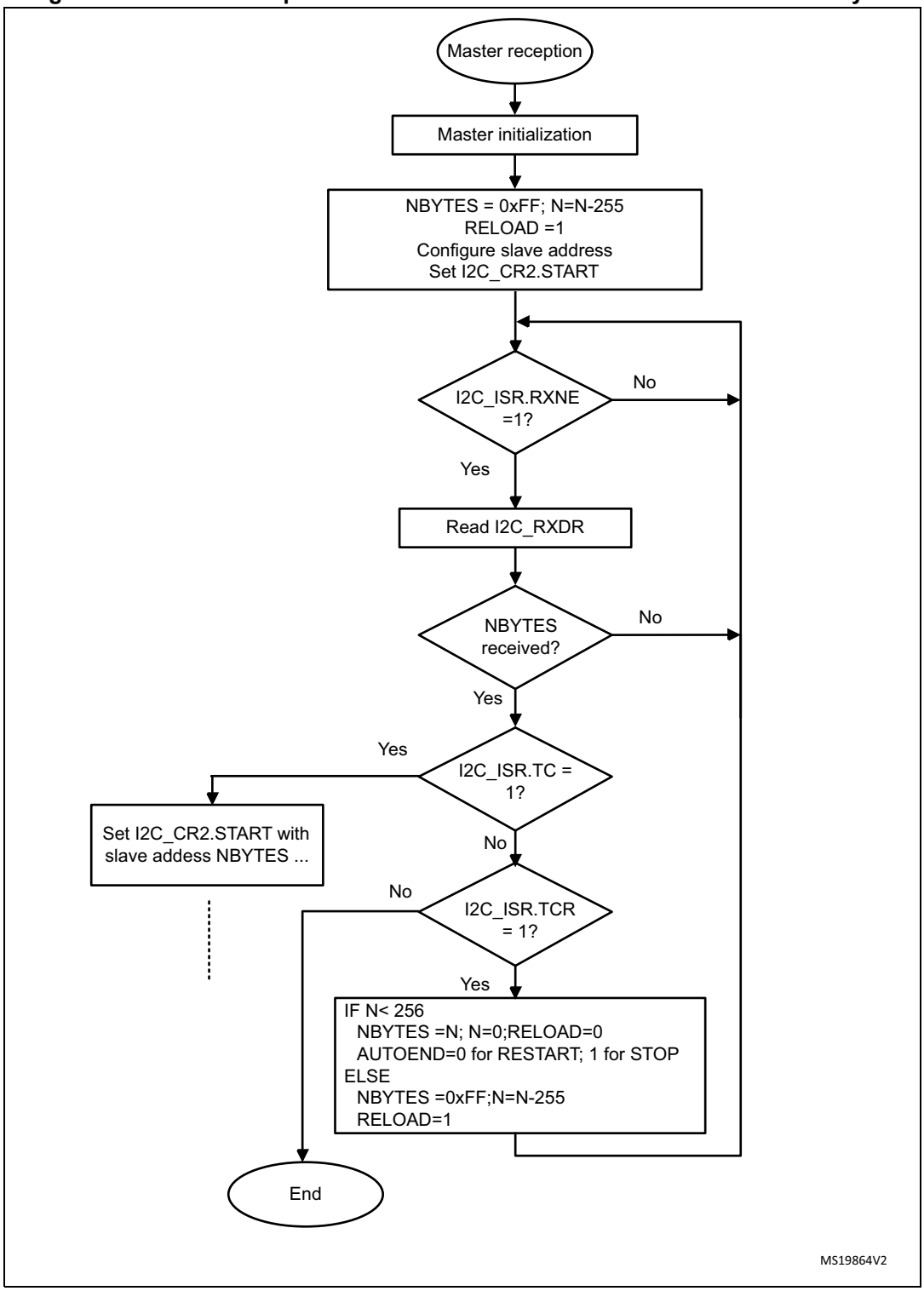
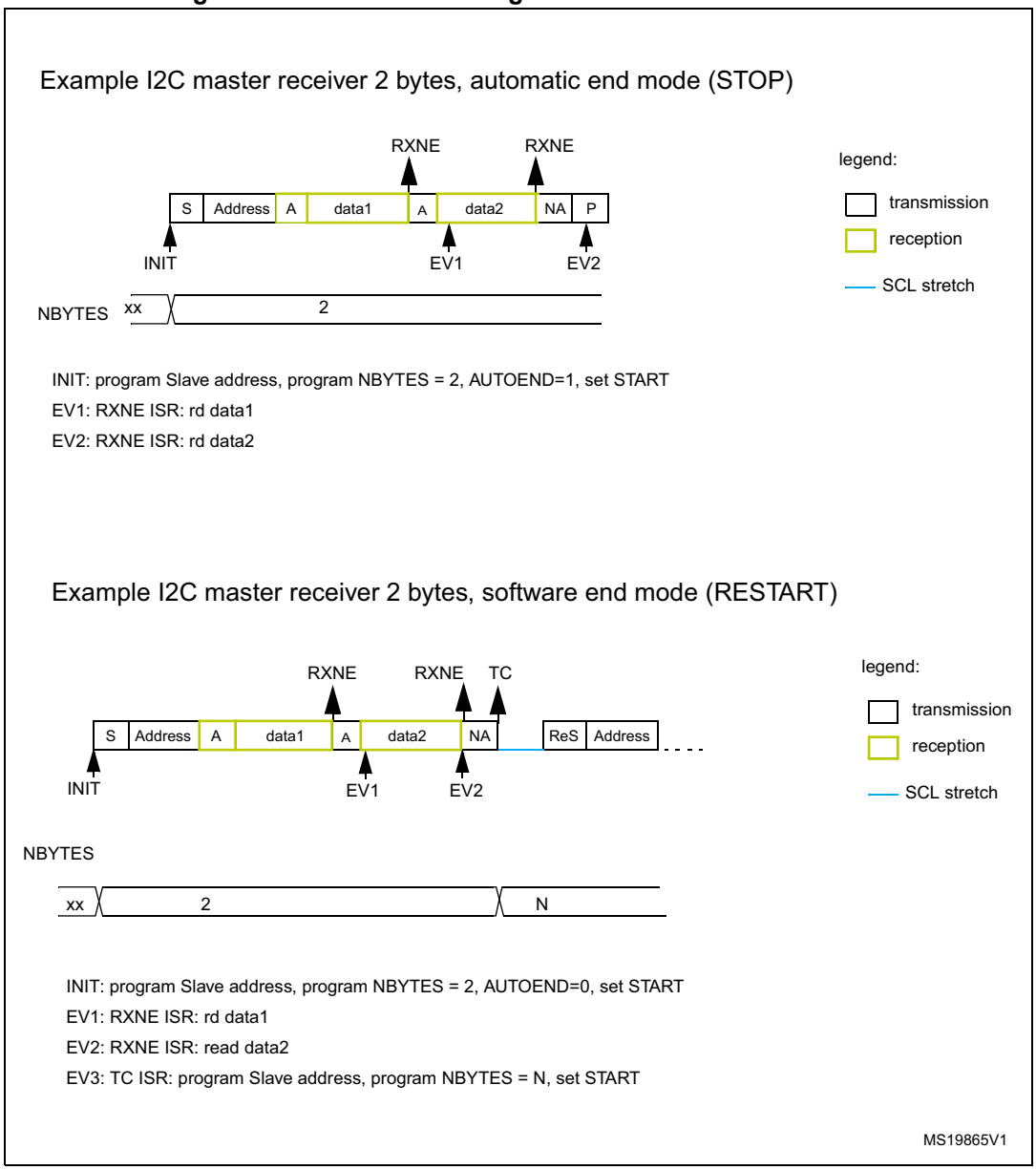


Figure 740. Transfer bus diagrams for I2C master receiver



43.4.10 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I^2C specification.

Table 481. Examples of timing settings for f_{I2CCLK} = 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)
	10 kHz	100 kHz	400 kHz
PRESC	1	1	0
SCLL	0xC7	0x13	0x9

Table 481. Examples of timing settings for f_{I2CCLK} = 8 MHz (continued)

Parameter	Standard-mode (Sm)		Fast-mode (Fm)
	10 kHz	100 kHz	400 kHz
t_{SCLL}	200x250 ns = 50 µs	20x250 ns = 5.0 µs	10x125 ns = 1250 ns
SCLH	0xC3	0xF	0x3
t_{SCLH}	196x250 ns = 49 µs	16x250 ns = 4.0µs	4x125ns = 500ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)
SDADEL	0x2	0x2	0x1
t_{SDADEL}	2x250 ns = 500 ns	2x250 ns = 500 ns	1x125 ns = 125 ns
SCLDEL	0x4	0x4	0x3
t_{SCLDEL}	5x250 ns = 1250 ns	5x250 ns = 1250 ns	4x125 ns = 500 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns

Table 482. Examples of timings settings for f_{I2CCLK} = 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)
	10 kHz	100 kHz	400 kHz
PRESC	3	3	1
SCLL	0xC7	0x13	0x9
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns
SCLH	0xC3	0xF	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)
SDADEL	0x2	0x2	0x2
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns
SCLDEL	0x4	0x4	0x3
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns

Table 483. Examples of timings settings for f_{I2CCLK} = 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLH	0xC3	0xF	0x3	0x1
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~875 ns(4)
SDADEL	0x2	0x2	0x3	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. The SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns
3. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns
4. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 250 ns

43.4.11 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBus specification.

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block

Read, Block Write and Block Write-Block Read Process Call. These protocols must be implemented by the user software.

For more details of these protocols, refer to SMBus specification.

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification.

Received Command and Data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to *Slave Byte Control mode* for more details.

Host Notify protocol

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus Host Address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the Alert Response Address.

When configured as a slave device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0.

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is

calculated by using the $C(x) = x_8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows a Not Acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

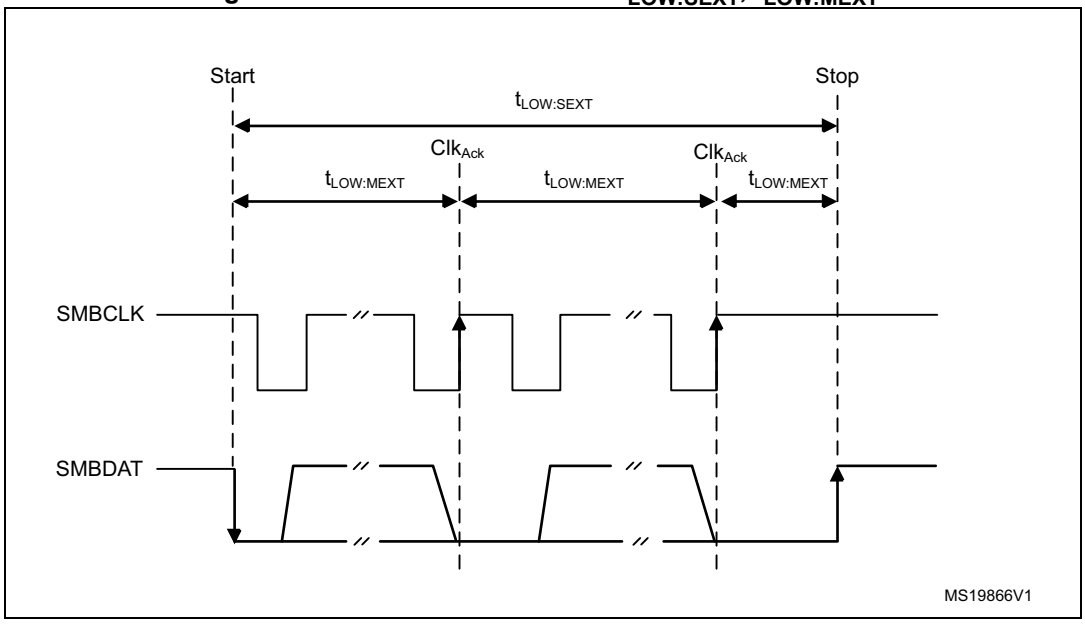
This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification.

Table 484. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (slave device)	-	25	ms
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (master device)	-	10	ms

1. $t_{LOW:SEXT}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2. $t_{LOW:MEXT}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

Figure 741. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for t_{IDLE} greater than $t_{HIGH,MAX}$. (refer to *Table 478: I2C-SMBUS specification data setup and hold times*).

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

43.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

Received Command and Data Acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to *Slave Byte Control mode* for more details.

Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to *Bus idle detection* for more details.

- The SMBus Device Default Address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus Host Address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The Alert Response Address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES-1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C is enabled.

Table 485. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Master Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
Master Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
Slave Tx/Rx with PEC	1	0	x	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- $t_{TIMEOUT}$ check

  In order to enable the $t_{TIMEOUT}$ check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the $t_{TIMEOUT}$ parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.

  Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register.

  If SCL is tied low for a time greater than (TIMEOUTA+1) x 2048 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 486: Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{TIMEOUT}$ = 25 ms)*.

Caution: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

  Depending on whether the peripheral is configured as a master or as a slave, the 12-bit TIMEOUTB timer must be configured in order to check $t_{LOW:SEXT}$ for a slave and $t_{LOW:MEXT}$ for a master. As the standard specifies only a maximum, the user can choose the same value for both.

  Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

  If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TIMEOUTB+1) x 2048 x t_{I2CCLK}, and in the timeout interval described in *Bus idle detection* section, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 487: Examples of TIMEOUTB settings for various I2CCLK frequencies*

Caution: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

Bus Idle detection

In order to enable the t_{IDLE} check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the t_{IDLE} parameter. The TIDLE bit must be configured to '1 in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than (TIMEOUTA+1) x 4 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

Refer to *Table 488: Examples of TIMEOUTA settings for various I2CCLK frequencies (max t_{IDLE} = 50 µs)*

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

43.4.13 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 486. Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{TIMEOUT}$ = 25 ms)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
32 MHz	0x186	0	1	391 x 2048 x 31.25 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

- Configuring the maximum duration of $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ to 8 ms:

Table 487. Examples of TIMEOUTB settings for various I2CCLK frequencies

f_{I2CCLK}	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 µs

Table 488. Examples of TIMEOUTA settings for various I2CCLK frequencies (max t_{IDLE} = 50 µs)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	t_{TIDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 µs

43.4.14 SMBus slave mode

This section is relevant only when SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

In addition to I2C slave transfer management (refer to *Section 43.4.8: I2C slave mode*) some additional software flowcharts are provided to support SMBus.

SMBus Slave transmitter

When the IP is used in SMBus, SBC must be programmed to '1' in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTES-1 and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES-1 data transfer.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 742. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC

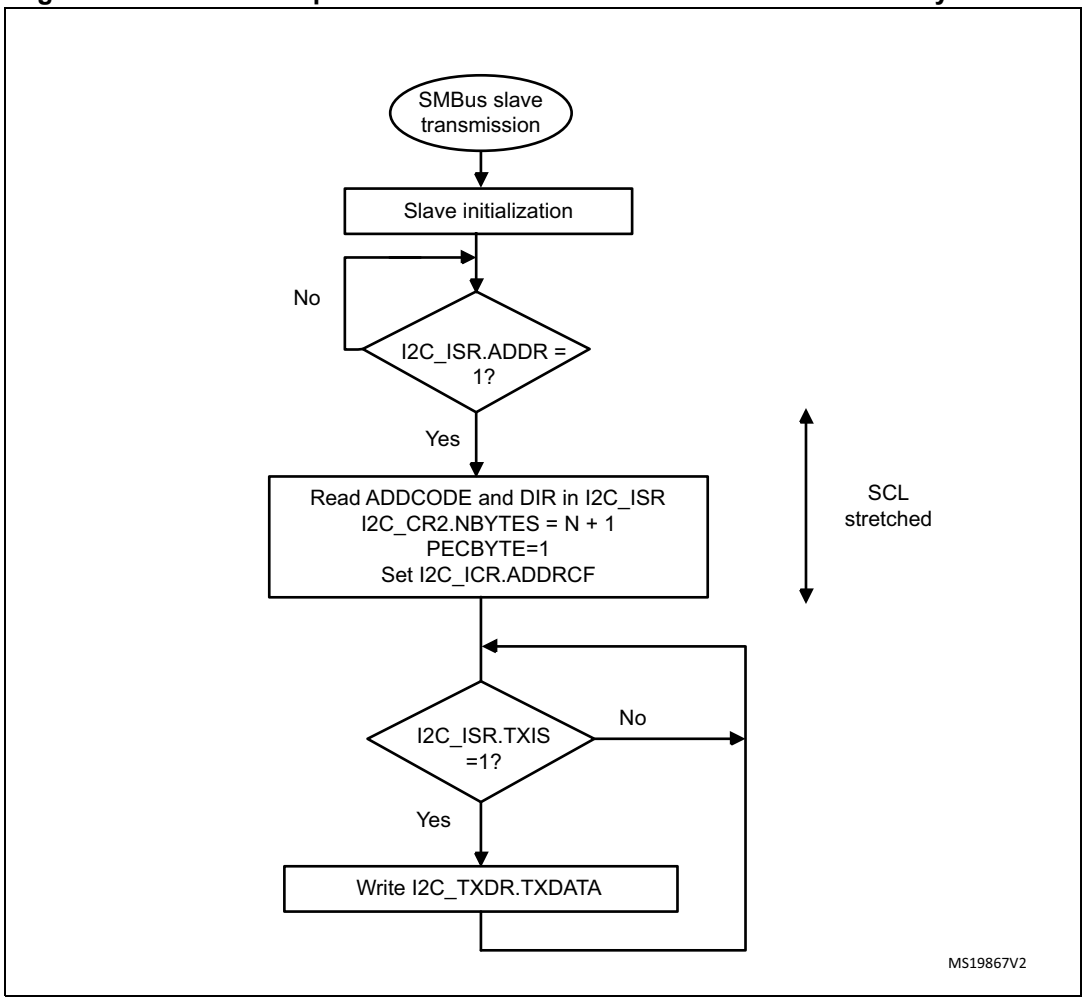
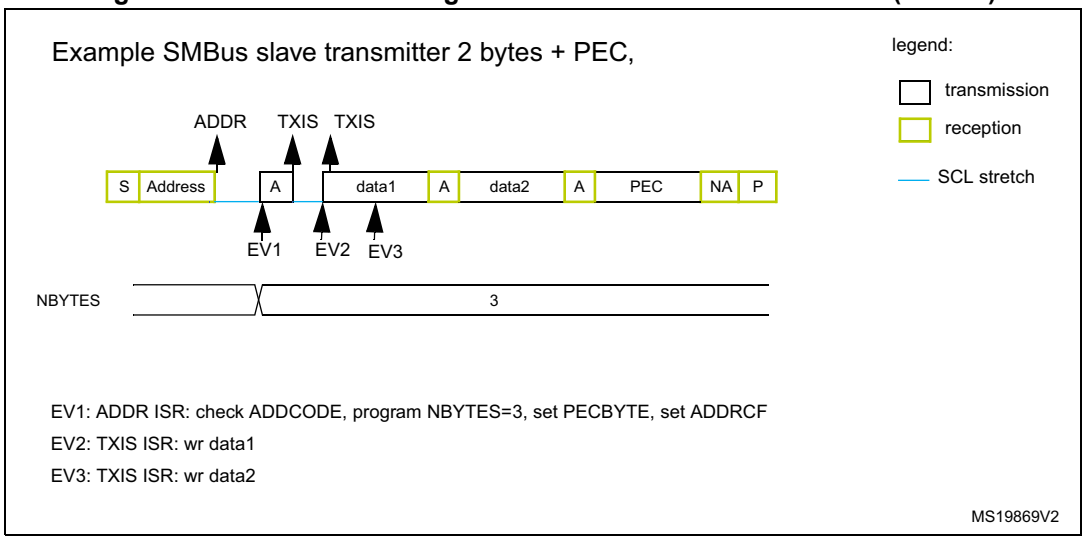


Figure 743. Transfer bus diagrams for SMBus slave transmitter (SBC=1)



SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to ‘1’ in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to *Slave Byte Control mode* for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES-1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, the user can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES-1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 744. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

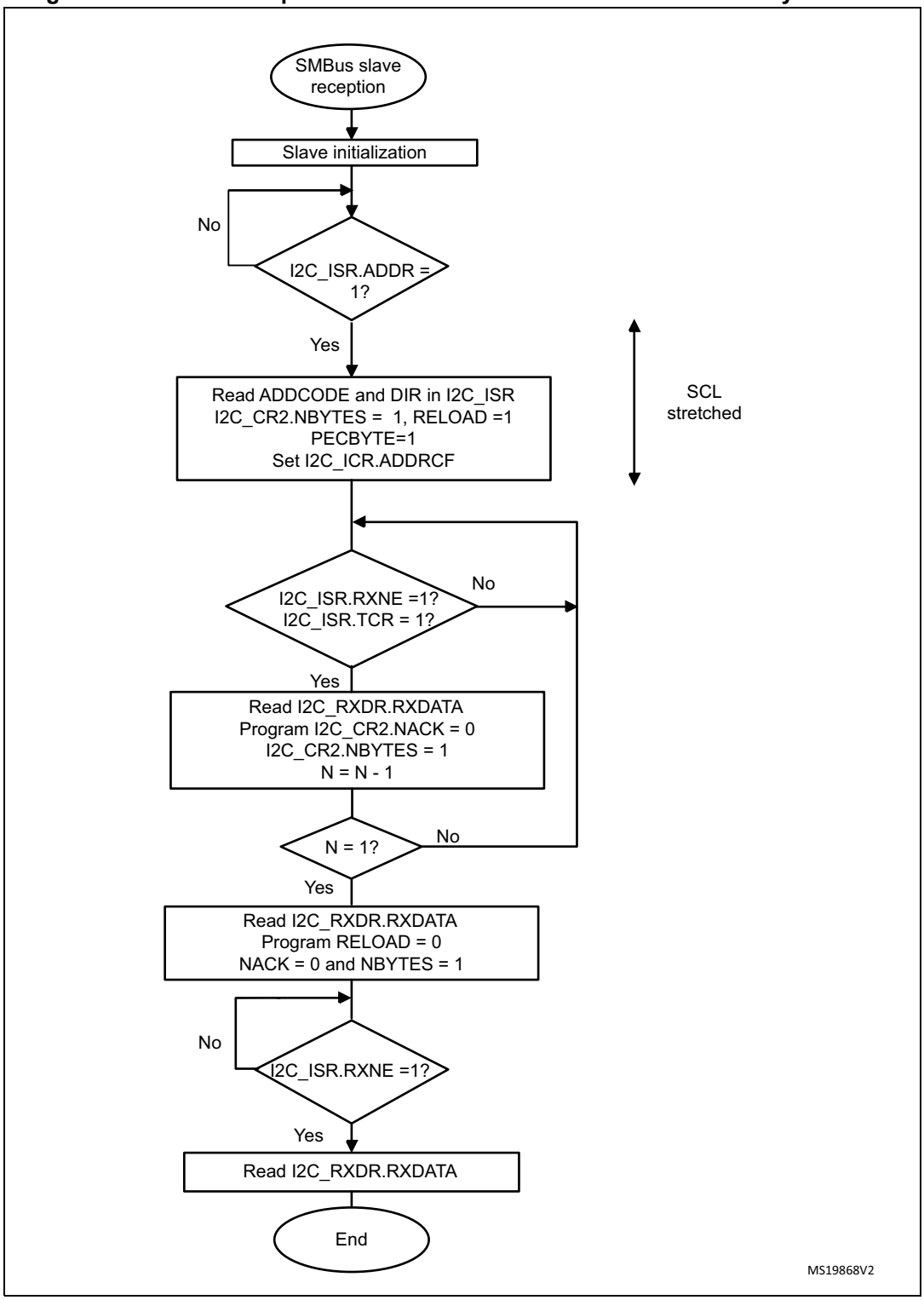
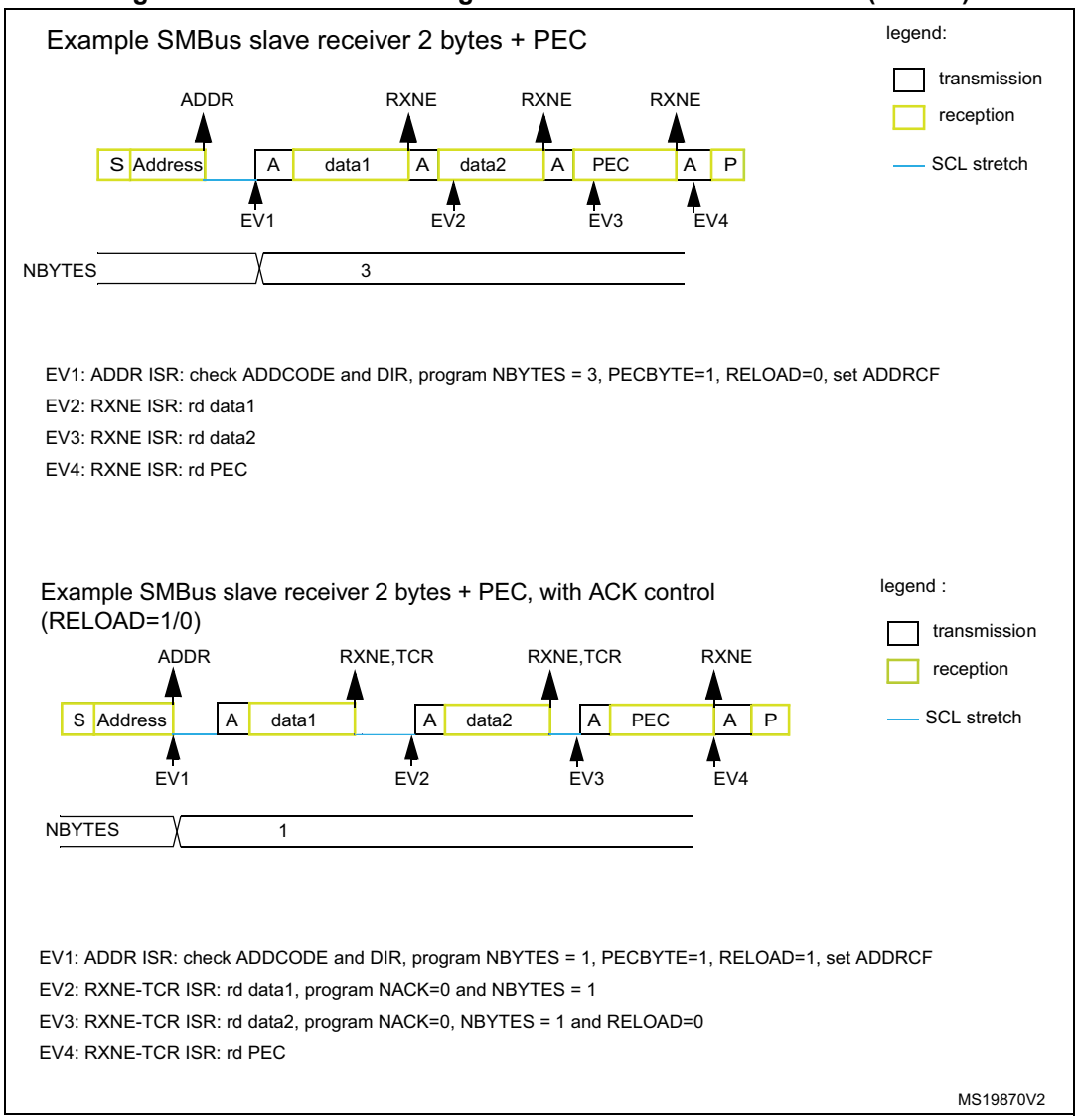


Figure 745. Bus transfer diagrams for SMBus slave receiver (SBC=1)



This section is relevant only when SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

In addition to I2C master transfer management (refer to *Section 43.4.9: I2C master mode*) some additional software flowcharts are provided to support SMBus.

SMBus Master transmitter

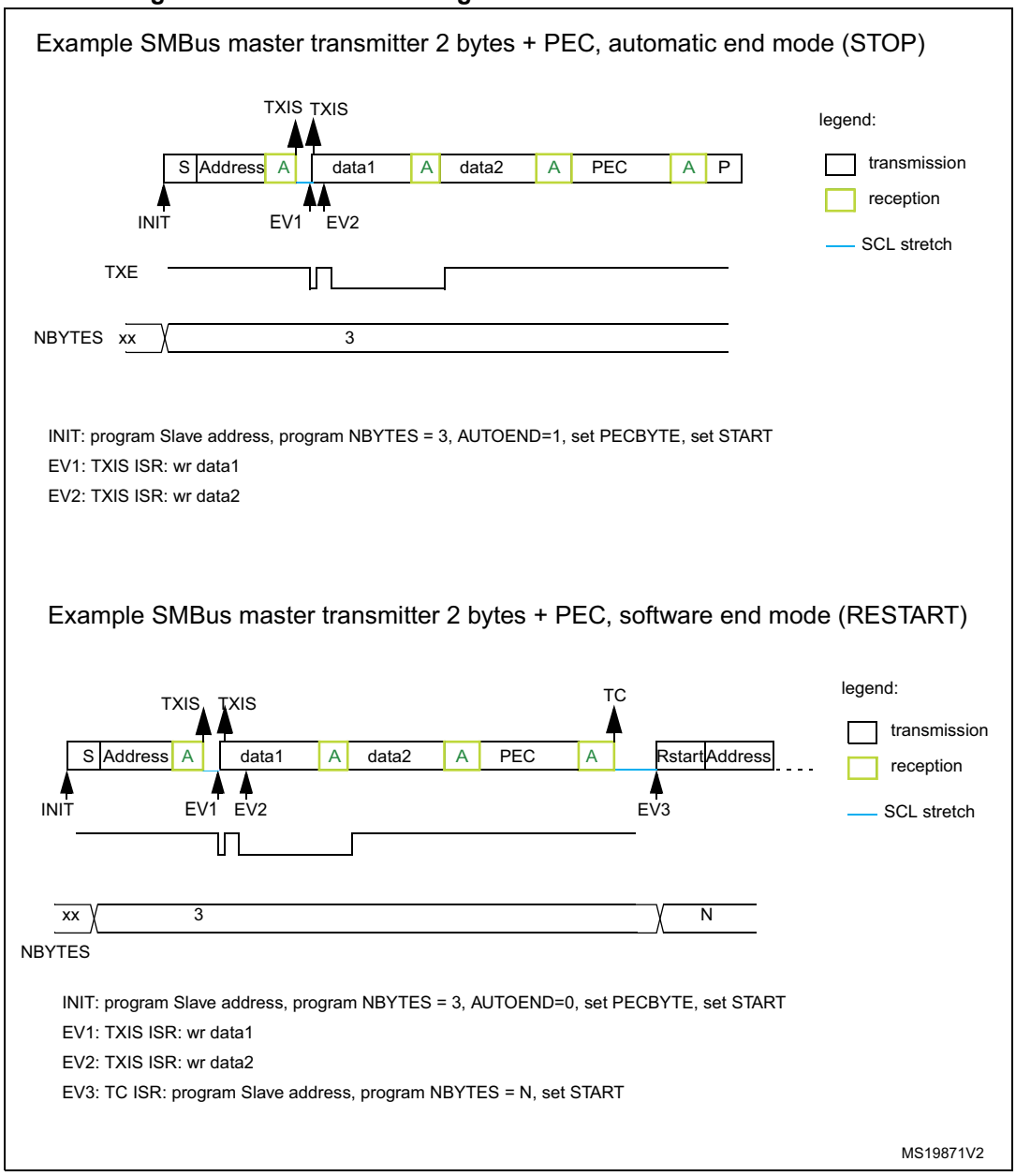
When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 746. Bus transfer diagrams for SMBus master transmitter



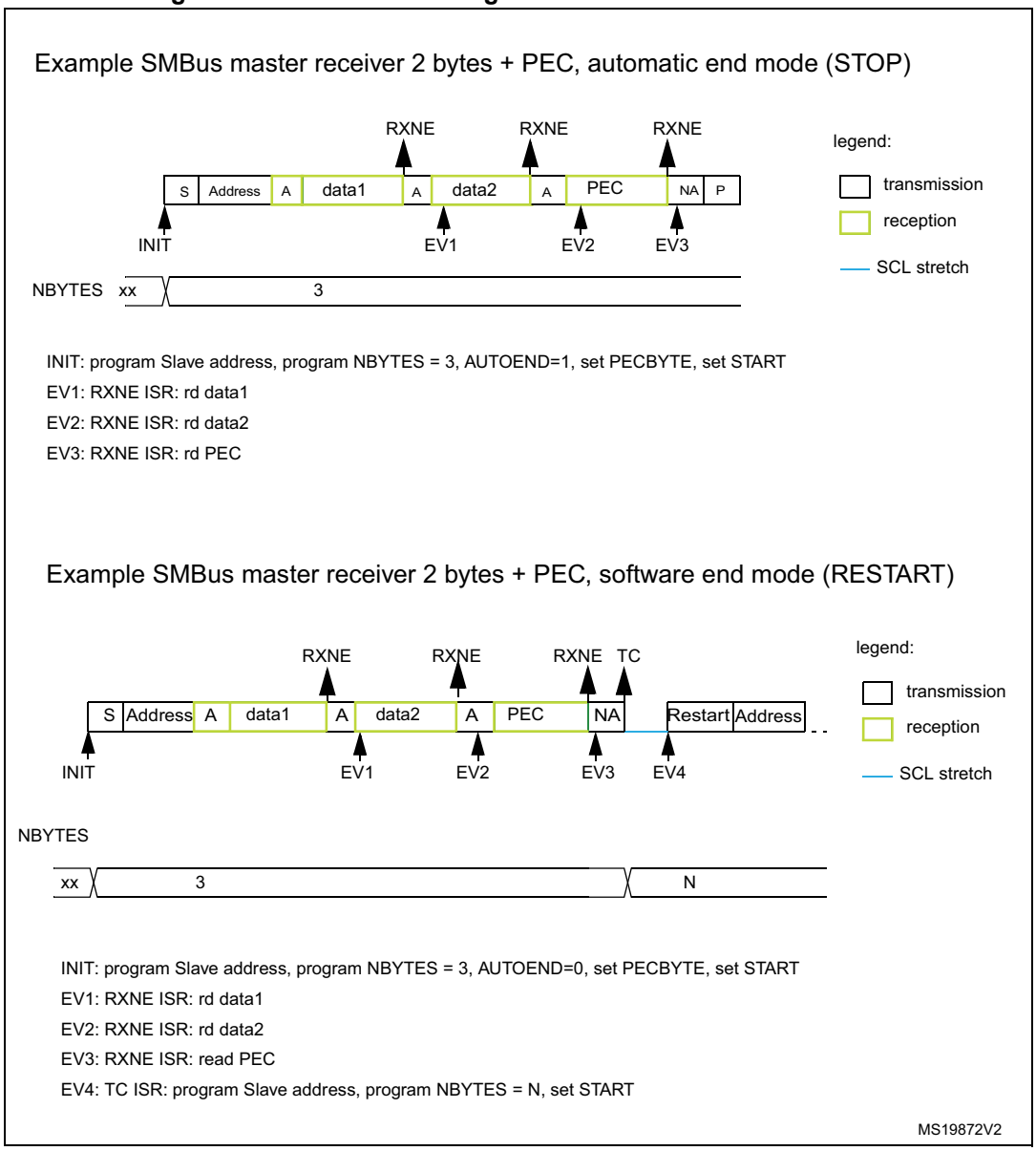
SMBus Master receiver

When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 747. Bus transfer diagrams for SMBus master receiver



43.4.15 Error conditions

The following errors are the error conditions which may cause communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (that is not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE=0, 0xFF is not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Packet error checking error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Timeout error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{LOW:MEXT}$ parameter).
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 43.3: I2C implementation*.

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

43.4.16 DMA requests

Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see *Chapter 23: Direct memory access controller (DMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Master transmitter*.

- In slave mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to *SMBus Slave transmitter* and *SMBus Master transmitter*.

Note: If DMA is used for transmission, the TXIE bit does not need to be enabled.

Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to *Chapter 23: Direct memory access controller (DMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.
- In slave mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see *Section 43.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to *SMBus Slave receiver* and *SMBus Master receiver*.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

43.4.17 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_ configuration bits in the DBG module.

43.5 I2C low-power modes

Table 489. Effect of low-power modes on the I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.

43.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 490. I2C Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes
		Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register	
		Stop detection interrupt flag	STOPF	STOPIE	Write STOPCF=1	
		Transfer Complete Reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0	
		Transfer complete	TC		Write START=1 or STOP=1	
		Address matched	ADDR	ADDRIE	Write ADDRCF=1	
		NACK reception	NACKF	NACKIE	Write NACKCF=1	
	I2C_ER	Bus error	BERR	ERRIE	Write BERRCF=1	Yes
		Arbitration loss	ARLO		Write ARLOCF=1	
		Overrun/ Underrun	OVR		Write OVRCF=1	
		PEC error	PECERR		Write PECERRCF=1	
		Timeout/t_{LOW} error	TIMEOUT		Write TIMEOUTCF=1	
		SMBus Alert	ALERT		Write ALERTCF=1	

43.7 I2C registers

Refer to *Section 1.2: Register conventions* for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

43.7.1 I2C memory map

Table 491. IC2 register memory map

Offset	Register name
0x00	*I2C control register 1 (I2C_CR1)*
0x04	*I2C control register 2 (I2C_CR2)*

Table 491. IC2 register memory map (continued)

Offset	Register name
0x08	*I2C own address 1 register (I2C_OAR1)*
0x0C	*I2C own address 2 register (I2C_OAR2)*
0x10	*I2C timing register (I2C_TIMINGR)*
0x14	*I2C timeout register (I2C_TIMEOUTR)*
0x18	*I2C interrupt and status register (I2C_ISR)*
0x1C	*I2C interrupt clear register (I2C_ICR)*
0x20	*I2C PEC register (I2C_PECR)*
0x24	*I2C receive data register (I2C_RXDR)*
0x28	*I2C transmit data register (I2C_TXDR)*

43.7.2 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBD EN	SMBH EN	GCEN	WUPE N	NOSTR ETCH	SBC
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF[3:0]				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **PECEN:** PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 22 **ALERTEN**: SMBus alert enable

0: The SMBus alert pin (SMBA) is not supported in host mode (SMBHEN=1). In device mode (SMBHEN=0), the SMBA pin is released and the Alert Response Address header is disabled (0001100x followed by NACK).
1: The SMBus alert pin is supported in host mode (SMBHEN=1). In device mode (SMBHEN=0), the SMBA pin is driven low and the Alert Response Address header is enabled (0001100x followed by ACK).

Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.
If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 21 **SMBDEN**: SMBus Device Default Address enable

0: Device Default Address disabled. Address 0b1100001x is NACKed.
1: Device Default Address enabled. Address 0b1100001x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 20 **SMBHEN**: SMBus Host Address enable

0: Host Address disabled. Address 0b0001000x is NACKed.
1: Host Address enabled. Address 0b0001000x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 19 **GCEN**: General call enable

0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wakeup from Stop mode enable

0: Wakeup from Stop mode disable.
1: Wakeup from Stop mode enable.

Note: If the Wakeup from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Note: WUPEN can be set only when DNF = '0000'

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.
0: Clock stretching enabled
1: Clock stretching disabled

Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 16 **SBC**: Slave byte control

This bit is used to enable hardware byte control in slave mode.
0: Slave byte control disabled
1: Slave byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable

0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to 1 t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to15 t_{I2CCLK}
Note: If the analog filter is also enabled, the digital filter is added to the analog filter.
This filter can only be programmed when the I2C is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generate an interrupt:
Arbitration Loss (ARLO)
Bus Error detection (BERR)
Overrun/Underrun (OVR)
Timeout detection (TIMEOUT)
PEC error detection (PECERR)
Alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer Complete interrupt enable
0: Transfer Complete interrupt disabled
1: Transfer Complete interrupt enabled
Note: Any of these events generate an interrupt:
Transfer Complete (TC)
Transfer Complete Reload (TCR)

Bit 5 **STOPIE**: Stop detection Interrupt enable
0: Stop detection (STOPF) interrupt disabled
1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (slave only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disable
1: Peripheral enable
Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

43.7.3 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte
This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE=0.
0: No PEC transfer.
1: PEC transmission/reception is requested
Note: Writing '0' to this bit has no effect.
This bit has no effect when RELOAD is set.
This bit has no effect is slave mode when SBC=0.
If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 25 **AUTOEND**: Automatic end mode (master mode)
This bit is set and cleared by software.
0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.
1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.
Note: This bit has no effect in slave mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode
This bit is set and cleared by software.
0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).
1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE=0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing '0' to this bit has no effect.

This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

In Master Mode:

0: No Stop generation.

1: Stop generation after current byte transfer.

Note: Writing '0' to this bit has no effect.

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCF bit in the I2C_ICR register.

0: No Start generation.

1: Restart/Start generation:

If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.

Otherwise setting this bit generates a START condition once the bus is free.

Note: Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10-bit slave address read sequence: Start + 2 bytes 10-bit address in write direction + Restart + 1st 7 bits of the 10-bit address in read direction.

1: The master only sends the 1st 7 bits of the 10-bit address, followed by Read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (master mode)
0: The master operates in 7-bit addressing mode,
1: The master operates in 10-bit addressing mode
Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (master mode)
0: Master requests a write transfer.
1: Master requests a read transfer.
Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Slave address (master mode)
In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] must be written with the 7-bit slave address to be sent. The bits SADD[9], SADD[8] and SADD[0] are don't care.
In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] must be written with the 10-bit slave address to be sent.
Note: Changing these bits when the START bit is set is not allowed.

43.7.4 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own Address 1 enable
0: Own address 1 disabled. The received slave address OA1 is NACKed.
1: Own address 1 enabled. The received slave address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own Address 1 10-bit mode
0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.
Note: This bit can be written only when OA1EN=0.

Bits 9:0 **OA1[9:0]**: Interface own slave address
7-bit addressing mode: OA1[7:1] contains the 7-bit own slave address. The bits OA1[9], OA1[8] and OA1[0] are don't care.
10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.
Note: These bits can be written only when OA1EN=0.

43.7.5 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own Address 2 enable
0: Own address 2 disabled. The received slave address OA2 is NACKed.
1: Own address 2 enabled. The received slave address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own Address 2 masks
000: No mask
001: OA2[1] is masked and don’t care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don’t care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don’t care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don’t care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don’t care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don’t care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don’t care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.
Note: These bits can be written only when OA2EN=0.
As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address
Note: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.

43.7.6 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to *I2C timings*) and for SCL high and low level counters (refer to *I2C master initialization*).

t_{PRESC} = (PRESC+1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

t_{SCLH} = (SCLH+1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

t_{SCLL} = (SCLL+1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C is disabled (PE = 0).

43.7.7 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable
0: Extended clock timeout detection is disabled
1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B
This field is used to configure the cumulative clock extension timeout:
In master mode, the master cumulative clock low extend time ($t_{LOW:MEXT}$) is detected
In slave mode, the slave cumulative clock low extend time ($t_{LOW:SEXT}$) is detected
$t_{LOW:EXT}$= (TIMEOUTB+1) x 2048 x t_{I2CCLK}
Note: These bits can be written only when TEXTEN=0.

Bit 15 **TIMOUTEN**: Clock timeout enable
0: SCL timeout detection is disabled
1: SCL timeout detection is enabled: when SCL is low for more than $t_{TIMEOUT}$ (TIDLE=0) or high for more than t_{IDLE} (TIDLE=1), a timeout error is detected (TIMEOUT=1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection
0: TIMEOUTA is used to detect SCL low timeout
1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)
Note: This bit can be written only when TIMOUTEN=0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A
This field is used to configure:
The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE=0
$t_{TIMEOUT}$= (TIMEOUTA+1) x 2048 x t_{I2CCLK}
The bus idle condition (both SCL and SDA high) when TIDLE=1
t_{IDLE}= (TIMEOUTA+1) x 4 x t_{I2CCLK}
Note: These bits can be written only when TIMOUTEN=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 43.3: I2C implementation.

43.7.8 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (Slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 **DIR**: Transfer direction (Slave mode)

This flag is updated when an address match event occurs (ADDR=1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected, or when PE=0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 11 **PECERR**: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 10 **OVR**: Overrun/Underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting *BERRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 7 **TCR**: Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE=0.

This flag is only for master mode, or for slave mode when the SBC bit is set.

Bit 6 **TC**: Transfer Complete (master mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE=0.

Bit 5 **STOPF**: Stop detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 4 **NACKF**: Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 3 **ADDR**: Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE=0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software when NOSTRETCH=1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN=1).

Note: This bit is cleared by hardware when PE=0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE=0.

43.7.9 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 11 **PECCF**: PEC Error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 43.3: I2C implementation.

Bit 10 **OVRCF**: Overrun/Underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

43.7.10 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 43.3: I2C implementation.

43.7.11 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]** 8-bit receive data
Data byte received from the I^2C bus

43.7.12 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data
Data byte to be transmitted to the I^2C bus
Note: These bits can be written only when TXE=1.

44 Independent watchdog (IWDG)

44.1 Introduction

The devices feature an embedded watchdog peripheral that offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral detects and solves malfunctions due to software failure, and triggers system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by RCOSC 16 MHz clock (safe clock) and thus stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to *Chapter 45: System window watchdog (WWDG)*.

44.2 IWDG main features

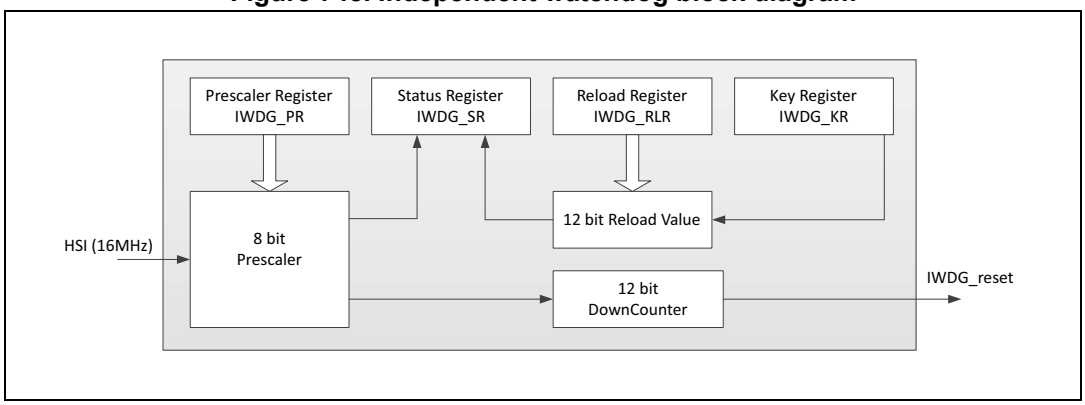
- Free-running downcounter
- Clocked from an independent RC oscillator
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

44.3 IWDG functional description

44.3.1 IWDG block diagram

Figure 748 shows the functional blocks of the independent watchdog module.

Figure 748. Independent watchdog block diagram



When the independent watchdog is started by writing the value 0x0000 CCCC in the *IWDG key register (IWDG_KR)*, the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

Once running, the IWDG cannot be stopped.

44.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than the value stored in the *IWDG window register (IWDG_WINR)*, then a reset is provided.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the *IWDG reload register (IWDG_RLR)* value and ease the cycle number calculation to generate the next reload.

Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically refreshes the counter value in the *IWDG reload register (IWDG_RLR)*.

Note: Writing the window value allows the counter value to be refreshed by the RLR when IWDG status register (IWDG_SR) is set to 0x0000 0000.

Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA).

44.3.3 Hardware watchdog

If the Hardware watchdog feature is enabled through UTEST MISCELLANEOUS IWDGx_EN bit (with x= 1 and 2), the watchdog is automatically enabled at power-on, and generates a reset unless the *IWDG key register (IWDG_KR)* is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

44.3.4 Register access protection

Write access to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)* and *IWDG window register (IWDG_WINR)* is protected. To modify them, the user must first write the code 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or of the downcounter reload value or of the window value is ongoing.

44.3.5 Debug mode

When the device enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on the configuration of the corresponding bit in DBGMCU freeze registers.

44.4 IWDG registers

Refer to *Section 1.2: Register conventions* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

44.4.1 IWDG memory map

Table 492. IWDG register memory map

Offset	Register name
0x00	*IWDG key register (IWDG_KR)*
0x04	*IWDG prescaler register (IWDG_PR)*
0x08	*IWDG reload register (IWDG_RLR)*
0x0C	*IWDG status register (IWDG_SR)*
0x10	*IWDG window register (IWDG_WINR)*

44.4.2 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 0x5555 to enable access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers (see *Section 44.3.4: Register access protection*)

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

44.4.3 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see *Section 44.3.4: Register access protection*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset in order to be able to change the prescaler divider.

000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

44.4.4 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Register access protection*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

44.4.5 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WVU	RVU	PVU
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five cycles).

Window value can be updated only when WVU bit is reset.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to five cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.

44.4.6 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 44.3.4*, they contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

45 System window watchdog (WWDG)

45.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications which require the watchdog to react within an accurate timing window.

45.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 750*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40.

45.3 WWDG functional description

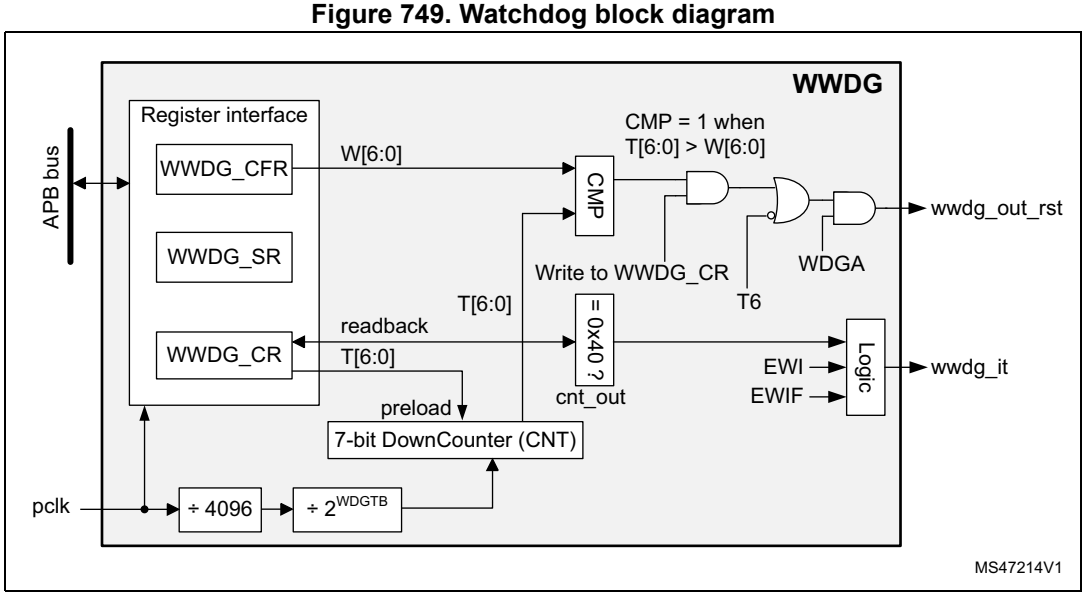
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 749* for the WWDG block diagram.

45.3.1 WWDG block diagram

Figure 749. Watchdog block diagram



45.3.2 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

45.3.3 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 750*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than the window register value and greater than 0x3F. *Figure 750* describes the window watchdog process.

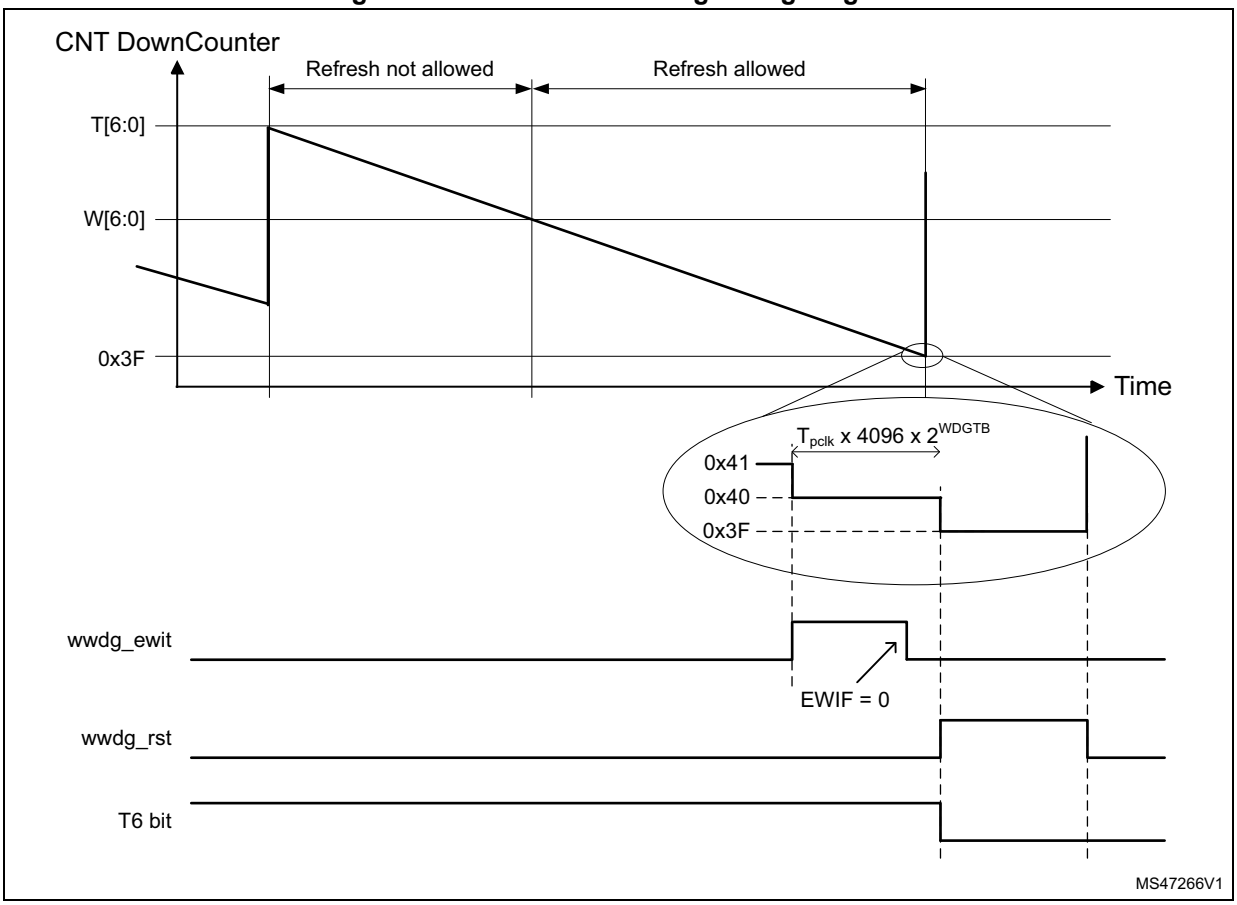
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

45.3.4 How to program the watchdog timeout

Use the formula in *Figure 750* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 750. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

t_{WWDG}: WWDG timeout

t_{PCLK}: APB clock period measured in ms

4096: value corresponding to internal divider

As an example, let's assume APB frequency is equal to 48 MHz, WDGTB[2:0] is set to 3 and T[5:0] is set to 63:

$$t_{WWDG} = (1/48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

45.3.5 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details refer to *Chapter 10: Debug support (DBG)*.

45.4 WWDG interrupts

The early wakeup interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the down-counter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) has to reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

Note: When the EWI interrupt cannot be served, for example due to a system lock in a higher priority task, the WWDG reset is eventually generated.

45.5 WWDG registers

Refer to *Section 1.2: Register conventions* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

45.5.1 WWDG memory map

Table 493. WWDG register memory map

Offset	Register name
0x000	*WWDG control register (WWDG_CR)*
0x004	*WWDG configuration register (WWDG_CFR)*
0x008	*WWDG status register (WWDG_SR)*

45.5.2 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

45.5.3 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK Counter Clock (PCLK div 4096) div 1

001: CK Counter Clock (PCLK div 4096) div 2

010: CK Counter Clock (PCLK div 4096) div 4

011: CK Counter Clock (PCLK div 4096) div 8

100: CK Counter Clock (PCLK div 4096) div 16

101: CK Counter Clock (PCLK div 4096) div 32

110: CK Counter Clock (PCLK div 4096) div 64

111: CK Counter Clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

45.5.4 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing ‘0’. Writing ‘1’ has no effect. This bit is also set if the interrupt is not enabled.

46 CAN subsystem

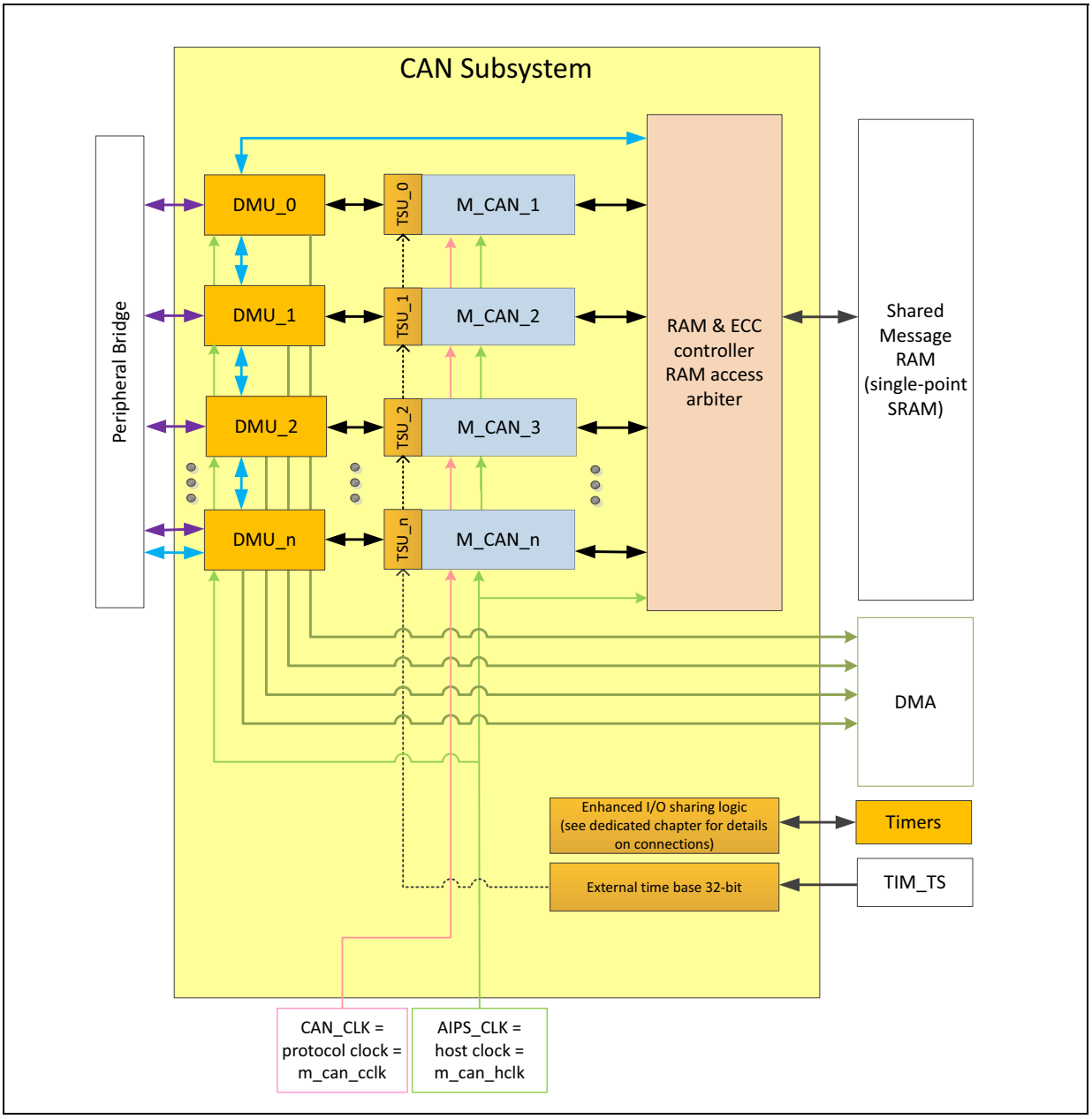
46.1 Introduction

SR5E1x has one CAN subsystem implemented (CAN Subsystem 0).

The Controller Area Network (CAN) subsystem consists of the modular CAN (M_CAN) modules and DMU and TSU for each M_CAN along with an integrated intelligent CAN RAM controller. The CAN RAM controller consists of additional logic for arbitration between the requests for the RAM access by the various CANs and CPU, ECC encoder/decoder for the Message RAM data and active transmit message buffer protection from CPU write access. The subsystem follows the little endian format.

The general CAN subsystem implementation is shown in *Section 46.2*. It interfaces with the Host processor bus using the peripheral bus. The RAM is not implemented inside the subsystem.

Figure 751. CAN subsystem generic block diagram



46.2 Features

The CAN subsystem consists of the following major blocks:

- Modular CAN cores: the registers of the CAN module can be accessed using the generic server interface (GSI)
- DMA interface unit: the registers of the CAN module can be accessed using the generic server interface (GSI)
- Timestamping unit: the registers of the TSU module can be accessed through DMA interface unit (DMU)
- CAN-RAM arbiter
- SRAM interface and memory organization
- ECC Controller

46.3 Controller area network (M_CAN)

Refer to *Chapter 47: Controller area network (M_CAN)*.

46.4 DMA interface unit (DMU)

Refer to *Chapter 48: DMA interface unit (DMU)*.

46.5 Timestamping unit (TSU)

Refer to *Chapter 49: Timestamping unit (TSU)*.

46.6 CAN RAM arbiter

This block acts as a dynamic arbiter between the various CAN nodes in one CAN subsystem and the CPU for granting access to the shared CAN memory. The dynamic arbitration ensures:

- 50% bandwidth for CPU, 50% shared by the CAN nodes when simultaneous access requested from all managers and CPU.
- Bandwidth dynamic upgrading to requesting managers if other managers do not request access.

Note: From the arbiter prospective the CPU bandwidth may be limited to 50%. The CPU may be waiting stated due to this bandwidth limitation when accessing the shared memory.

46.6.1 Features

- Active low asynchronous reset
- 1–CPU + up to arbiter.
- Combined dynamic arbitration scheme: bandwidth between CPU and M_CAN interfaces can be redistributed. If active, the CPU interface is granted a 50% portion of the available bandwidth.
- Bandwidth dynamic upgrading to requesting managers if other managers do not request access.
- The SRAM access by a single manager needs two cycles (address and data) for each read/write command. The arbiter has a pseudo address pre-fetching mechanism. This mechanism acts when multiple managers are accessing the SRAM and it overlaps the address cycle of one manager with the data cycle of another manager. In case of multiple managers accessing the SRAM, the pseudo address pre-fetching scheme saves multiple clock cycles.
- The peripheral GSI module enable acts as a request from each manager. Each manager waits at least one clock cycle before the grant is given. The arbiter ensures that the CPU does not wait for more than 1 clock cycle, hence ensuring a 50% guaranteed bandwidth.
- Time slot is defined as one peripheral clock cycle.

46.6.2 Functional overview using examples

The dynamic arbitration scheme is explained using the following examples.

- Example 1 (active CPU IF, 3 out of 3 CAN nodes are active):
  - CPU obtains every 2nd slot
  - CAN 1/2/3 obtains every 6th time slot
- Example 2 (inactive CPU IF, 4 out of 4 CAN nodes are active):
  - CPU obtains no time slots
  - CAN 1/2/3/4 obtains every 4th time slot
- Example 3 (active CPU IF, 2 out of 4 CAN node are active):
  - CPU obtains every 2nd time slot
  - Active CAN 1/2 obtains every 4th time slot
- Example 4 (inactive CPU IF, 2 out of 4 CAN node are active):
  - CPU obtains no time slots
  - Active CAN 1/2 obtains every 2nd time slot

46.7 SRAM interface and memory organization

The CAN subsystem interfaces with an external RAM using this interface. The RAM sizes required for the CAN subsystem are detailed in *Section 46.11: Shared memory map*.

46.7.1 ECC controller

The RAM embeds ECC logic and code to provide Single Error Correction / Double Error Detection (SECCDED). The module does not guarantee any proper functionality if more than 2 bits are in errors. It supports only 32-bit write access and 8-/16-/32-bit read access.

The ECC error is reported to each CAN module as well as available at the CAN subsystem top, so that the same can be forwarded to the Error Management Module.

The module uses Hamming code for single error correction and double error detection logic. It involves transmitting data with multiple ecc_check bits and decoding the associated ecc_check bits when receiving data to detect errors. The SECDED Hamming code is not able to detect three bit errors. Rather, in the presence of a three-bit error, a conventional SECDED code returns an error code that is indistinguishable from an error code resulting from a single bit error. Hence, this module does not guarantee any proper functionality when more than 2 bits are in error.

The ecc_check bits are parallel parity bits generated from XORing certain bits in the original data word. If bit error(s) are introduced in the codeword, several ecc_check bits show parity errors after decoding the retrieved codeword. The combination of these ecc_check bit errors displays the nature of the error. In addition, the position of any single bit error is identified from the ecc_check bits.

The ECC error address is reported at the top of the subsystem using a 16-bit ECC error address signal. This address is valid only when the ECC bit error output is valid.

46.7.1.1 Features

The ECC module supports the following:

- ecc_error = ‘00’

  No error is detected in the read data from memory.
- ecc_error = ‘01’

  This indicates single bit error occurred within the 39-bit codeword. In addition, the error is corrected, and the forwarded data to the manager is error free.
- ecc_error = ‘10’

  This indicates that a 2-bit error has occurred within the codeword. In this case, no error correction is possible.
- ecc_error = ‘11’

  This indicates that errors beyond the detection capability have occurred within the codeword and no error correction is possible. This is an invalid error type.

46.8 CAN nodes

Table 494 lists the different CAN nodes for the CAN subsystem.

Table 494. CAN bus controllers

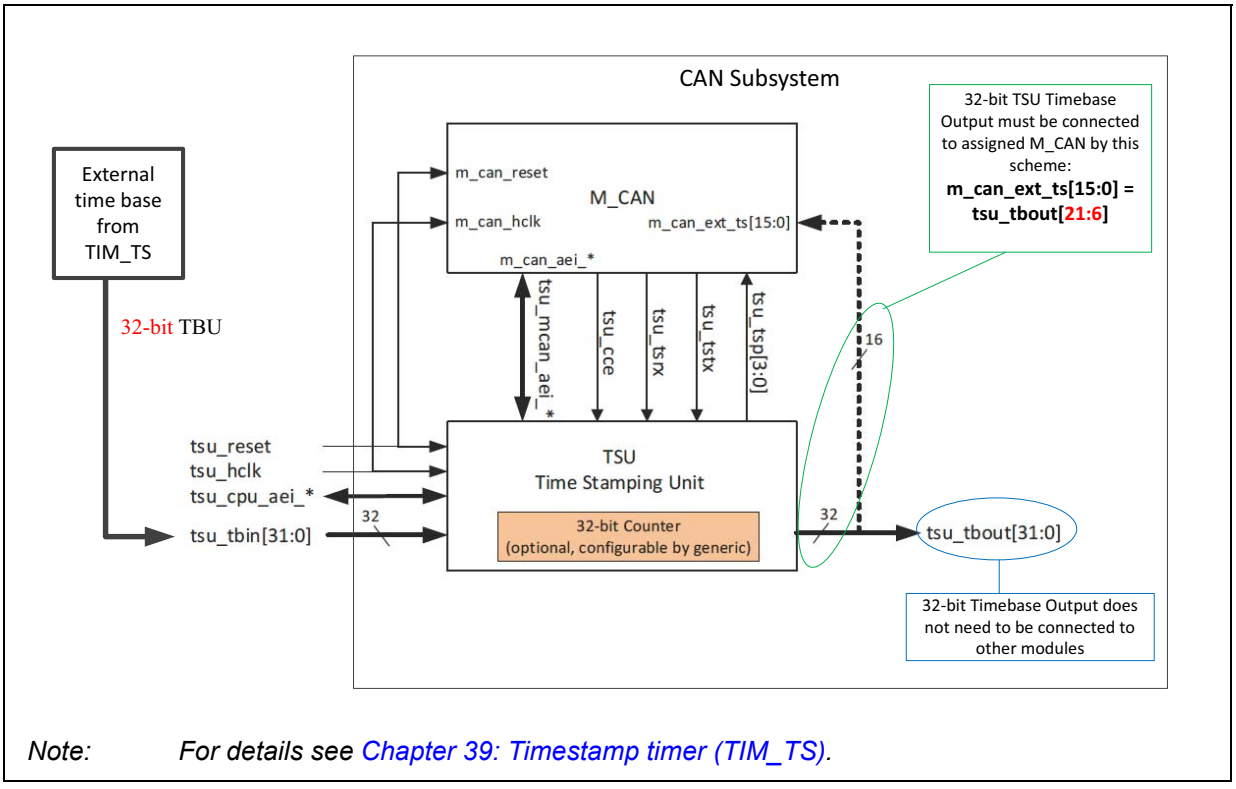
Subsystem	Peripheral bridge instance	Shared message RAM size (Kbyte)	Type	Instance	CAN FD support up to 64 data byte(1)	Debug on CAN support
CAN subsystem 0	APB2	16	M_CAN	CAN_SUB_0_M_CAN_1	Yes	Yes
				CAN_SUB_0_M_CAN_2	Yes	Yes
				CAN_SUB_0_M_CAN_3	Yes	No
				CAN_SUB_0_M_CAN_4	Yes	No

1. M_CAN revision 3.3.0.

46.9 External Timestamp

The implementation of the external timestamping with TSU modules is depicted in *Figure 752: CAN external time base*. The time stamp bus width has to be 32-bit wide.

Figure 752. CAN external time base



Note: For details see Chapter 39: Timestamp timer (TIM_TS).

46.10 External signal description

The CAN subsystem has the following external pins.

- M_CAN_x_RX: M_CAN x receive input
- M_CAN_x_TX: M_CAN x transmit output

46.11 Shared memory map

Table 495 shows the shared memory map vs CAN subsystem transmit, receive and filters elements.

Table 495. Shared memory map versus CAN transmit, receive and filters elements

<table>
<tr><th>Start Address Offset (byte)</th><th>End Address Offset (byte)</th><th colspan="2">CAN block/sub block</th><th>Comment</th></tr>
<tr><td colspan="5">CAN Subsystem_0 - APB2</td></tr>
<tr><td rowspan="7">0x0000</td><td rowspan="7">0x3FFF</td><td rowspan="7">CAN_SUB_0_M_CAN_1
CAN_SUB_0_M_CAN_2
CAN_SUB_0_M_CAN_3
CAN_SUB_0_M_CAN_4</td><td>Standard Filters</td><td rowspan="7">This available physical memory space can be configured for Filters/ FIFO/Buffer.

This memory space can be flexible, assigned to any of the implemented M_CAN modules.</td></tr>
<tr><td>Extended Filters</td></tr>
<tr><td>Rx FIFO 0</td></tr>
<tr><td>Rx FIFO 1</td></tr>
<tr><td>Rx Buffer</td></tr>
<tr><td>Tx Event FIFO</td></tr>
<tr><td>Tx Buffers</td></tr>
</table>

Note: There is no write access protection from a CAN module to the message memory space.

The start address for every memory section is to be configured by the user in the relevant CAN modules registers. However there is no hardware check in the RAM interface for the start address consistency.

However the number of filters, and the Tx/Rx buffers and the triggers used, it is mandatory for the user to configure the start address registers, specified in Table 496, in order to fit the memory mapping specified in Table 495.

The CPU uses the peripheral interface to access the memory locations. It must be noted that there is no transfer error reported to the CPU in case of access to reserved memory locations.

Table 496. Detailed mapping in CAN sub-blocks

CAN sub-block	Element name	Comment
Standard Filters	Standard Message ID filter 0 ... Standard Message ID filter 127	address configured in SIDFC [FLSSA]
Extended Filters	Extended Message ID filter 0 ... Extended Message ID filter 63	address configured in XIDFC [FLESA]
Rx FIFO0	Rx FIFO0 element 0 ... Rx FIFO0 element 63	address configured in RXF0C[F0SA]
Rx FIFO1	Rx FIFO1 element 0 ... Rx FIFO1 element 63	address configured in RXF1C[F1SA]
Rx Buffer	Rx Buffer element 0 ... Rx Buffer element 63	address configured in RXBC[RBSA]
Tx Event FIFO	Tx event FIFO element 0 ... Tx event FIFO element 31	address configured in TXEFC[EFSA]
Tx Buffers	Tx Buffer element 0 ... Tx Buffer element 31	address configured in TXBC[TBSA]

47 Controller area network (M_CAN)

47.1 Overview

Table 497. IP user manual versions

IP name	Version
M_CAN IP	Revision 3.3.0

M_CAN functionality conforms to CAN specification V2.0B active for each M_CAN node.The M_CAN performs communication according to the CAN protocol specification 2.0 part A,B and to CAN FD 1.0. Flexible assignment of message objects to nodes.
The bit rate can be programmed to values up to 1 Mbit/s for standard CAN frames. High bit rates are possible in CAN FD mode. Additional transceiver hardware is required for connection to the physical layer.

Note: Implemented CAN nodes are fully compliant to ISO CAN FD specification ISO 11898-1: 2015.

The message storage is intended to be a single-ported message RAM outside of the module. It is connected to the M_CAN via the generic manager interface. Depending on the chosen device, multiple M_CAN controllers can share the same message RAM.

All functions concerning the handling of messages are implemented by the Rx handler and the Tx handler. The Rx handler manages message acceptance filtering, the transfer of received messages from the CAN core to the message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the message RAM to the CAN core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_CAN’s clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

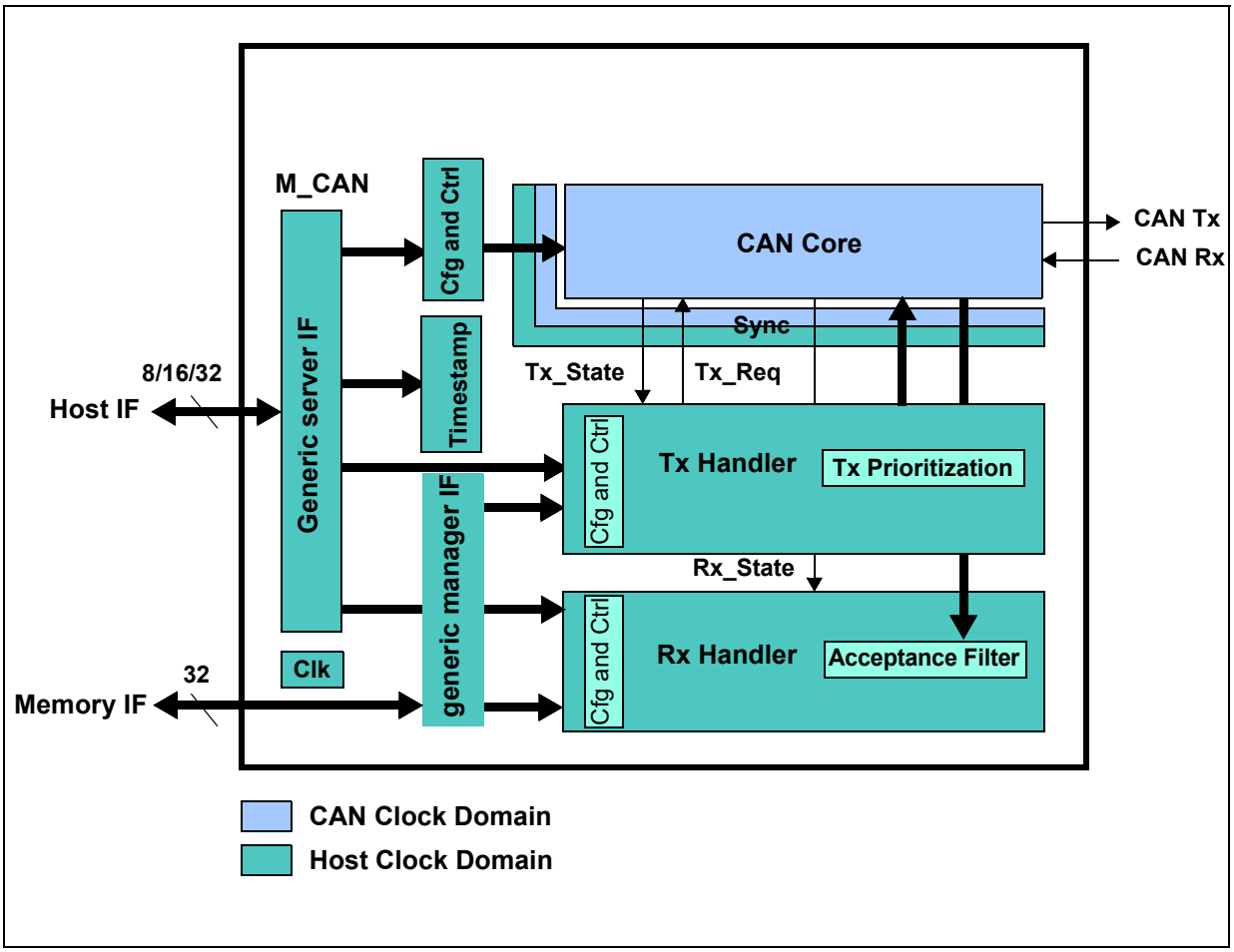
47.1.1 Features

The following are the features of Modular CAN cores.

- Conforms with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015
- CAN Flexible data-rate (ISO CAN FD) protocol with 64 data bytes is supported
- Bit rates up to 1 Mbit/s in standard CAN mode
- Bit rates up to 8 Mbit/s in ISO CAN FD mode
- CAN error logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of high priority messages
- Up to 64 dedicated receive buffers
- Up to 32 dedicated transmit buffers
- Configurable transmit FIFO
- Configurable transmit Queue
- Configurable transmit Event FIFO
- Direct message RAM access for Host CPU
- Multiple M_CANs share the same message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8-/16-/32-bits generic server interface for connection customer-specific host CPUs
- Two clock domains (CAN clock and host clock)
- Power-down support
- DMA support (DMU)
- Support of hardware timestamping according to CiA 603 (TSU)

47.1.2 Block diagram

Figure 753. M_CAN core block diagram



- CAN core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Clk: synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- Cfg and Ctrl: CAN core related configuration and control bits.
- Interrupt and timestamp: interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: controls the message transfer from the external Message RAM to the CAN core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.
- Rx Handler: controls the transfer of received messages from the CAN core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored

together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

- Generic server interface: connects the M_CAN to a customer specific Host CPU. The Generic server interface is capable of connecting to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- Generic manager interface: connects the M_CAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB × 32 bits.

47.1.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN core, a separate clock without any modulation has to be provided as CAN clock.

Within the M_CAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note: In order to achieve a stable function of the M_CAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

Caution: Stop this module when the device is in STOP mode. If CAN message reception is mandatory, check that the PLL is unlocked before transmitting any message. Use the external oscillator (XOSC) instead of the internal RC oscillator (IRCOSC) during the STOP mode.

47.1.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to EINT0 or to EINT1. By default all interrupts are routed to interrupt line EINT0. By programming EINT0 and EINT1 bits of the Interrupt Line Enable (ILE) register, the interrupt lines can be enabled or disabled separately.

47.2 Memory map and register descriptions

47.2.1 Hardware reset description

After hardware reset, the registers of the M_CAN hold the reset values listed in *Table 498*. Additionally the Bus_Off state is reset and the M_CAN Tx output is set to recessive (HIGH). The value 0x0001 (bit 0 of CCCR register CCCR[INIT] = 1) in the CC Control Register enables software initialization. The M_CAN does not influence the CAN bus until the CPU resets bit 0 of CCCR register CCCR[INIT] = 0.

47.2.2 Register map

The M_CAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic server interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

Note: Write access by the Host CPU to registers/bits marked with "P = Protected Write" is possible only when the bit 1 [CCE] and bit 0 [INIT] of the CCCR register are set to 1.

There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

Caution: Any write access to reserved or not implemented registers in the 16 KB slot assigned by the peripherals bridge to the M_CAN IP does not generate any bus access error.

Table 498. M_CAN memory map

Address Offset	Register Name	Access(1)	Reset value	Location
0x0000	Core Release Register (CREL)	R	0x33081114	*Section 47.2.3.1*
0x0004	Endian Register (ENDN)	R	0x87654321	*Section 47.2.3.2*
0x0008–0x000B	Reserved			
0x000C	Data Bit Timing and Prescaler Register (DBTP)	RP	0x00000A33	*Section 47.2.3.3*
0x0010	Test Register (TEST)	RP	0x00000000	*Section 47.2.3.4*
0x0014	RAM Watchdog Register (RWD)	RP	0x00000000	*Section 47.2.3.5*
0x0018	CC Control Register (CCCR)	RWPp	0x00000001	*Section 47.2.3.6*
0x001C	Nominal Bit Timing and Prescaler Register (NBTP)	RP	0x06000A03	*Section 47.2.3.7*
0x0020	Timestamp Counter Configuration Register (TSCC)	RP	0x00000000	*Section 47.2.3.8*
0x0024	Timestamp Counter Value Register (TSCV)	RC	0x00000000	*Section 47.2.3.9*
0x0028	Timeout Counter Configuration Register (TOCC)	RP	0xFFFF0000	*Section 47.2.3.10*
0x002C	Timeout Counter Value Register (TOCV)	RC	0x0000FFFF	*Section 47.2.3.11*
0x0030–0x003F	Reserved			
0x0040	Error Counter Register (ECR)	RX	0x00000000	*Section 47.2.3.12*
0x0044	Protocol Status Register (PSR)	RXS	0x00000707	*Section 47.2.3.13*
0x0048	Transmitter Delay Compensation Register (TDCR)	RP	0x00000000	*Section 47.2.3.14*
0x004C–0x004F	Reserved			
0x0050	Interrupt Register (IR)	R/W	0x00000000	*Section 47.2.3.15*
0x0054	Interrupt Enable Register (IE)	R/W	0x00000000	*Section 47.2.3.16*
0x0058	Interrupt Line Select Register (ILS)	R/W	0x00000000	*Section 47.2.3.17*
0x005C	Interrupt Line Enable Register (ILE)	R/W	0x00000000	*Section 47.2.3.18*
0x0060–0x007F	Reserved			
0x0080	Global Filter Configuration Register (GFC)	RP	0x00000000	*Section 47.2.3.19*
0x0084	Standard ID Filter Configuration Register (SIDFC)	RP	0x00000000	*Section 47.2.3.20*
0x0088	Extended ID Filter Configuration Register (XIDFC)	RP	0x00000000	*Section 47.2.3.21*
0x008C–0x008F	Reserved			

Table 498. M_CAN memory map (continued)

Address Offset	Register Name	Access(1)	Reset value	Location
0x0090	Extended ID AND Mask Register (XIDAM)	RP	0x1FFFFFFF	*Section 47.2.3.22*
0x0094	High Priority Message Status Register (HPMS)	R	0x00000000	*Section 47.2.3.23*
0x0098	New Data 1 Register (NDAT1)	R/W	0x00000000	*Section 47.2.3.24*
0x009C	New Data 2 Register (NDAT2)	R/W	0x00000000	*Section 47.2.3.25*
0x00A0	Rx FIFO 0 Configuration Register (RXF0C)	RP	0x00000000	*Section 47.2.3.26*
0x00A4	Rx FIFO 0 Status Register (RXF0S)	R	0x00000000	*Section 47.2.3.27*
0x00A8	Rx FIFO 0 Acknowledge Register (RXF0A)	R/W	0x00000000	*Section 47.2.3.28*
0x00AC	Rx Buffer Configuration Register (RXBC)	RP	0x00000000	*Section 47.2.3.29*
0x00B0	Rx FIFO 1 Configuration Register (RXF1C)	RP	0x00000000	*Section 47.2.3.30*
0x00B4	Rx FIFO 1 Status Register (RXF1S)	R	0x00000000	*Section 47.2.3.31*
0x00B8	Rx FIFO 1 Acknowledge Register (RXF1A)	R/W	0x00000000	*Section 47.2.3.32*
0x00BC	Rx Buffer / FIFO Element Size Configuration Register (RXESC)	RP	0x00000000	*Section 47.2.3.33*
0x00C0	Tx Buffer Configuration Register (TXBC)	RP	0x00000000	*Section 47.2.3.34*
0x00C4	Tx FIFO/Queue Status Register (TXFQS)	R	0x00000000	*Section 47.2.3.35*
0x00C8	Tx Buffer Element Size Configuration Register (TXESC)	RP	0x00000000	*Section 47.2.3.36*
0x00CC	Tx Buffer Request Pending Register (TXBRP)	R	0x00000000	*Section 47.2.3.37*
0x00D0	Tx Buffer Add Request Register (TXBAR)	R/W	0x00000000	*Section 47.2.3.38*
0x00D4	Tx Buffer Cancellation Register (TXBCR)	R/W	0x00000000	*Section 47.2.3.39*
0x00D8	Tx Buffer Transmission Occurred Register (TXBTO)	R	0x00000000	*Section 47.2.3.40*
0x00DC	Tx Buffer Cancellation Finished Register (TXBCF)	R	0x00000000	*Section 47.2.3.41*
0x00E0	Tx Buffer Transmission Interrupt Enable (TXBTIE)	R/W	0x00000000	*Section 47.2.3.42*
0x00E4	Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)	R/W	0x00000000	*Section 47.2.3.43*
0x00E8–0x00EF	Reserved			
0x00F0	Tx Event FIFO Configuration Register (TXEFC)	RP	0x00000000	*Section 47.2.3.44*
0x00F4	Tx Event FIFO Status Register (TXEFS)	R	0x00000000	*Section 47.2.3.45*
0x00F8	Tx Event FIFO Acknowledge Register (TXEFA)	R/W	0x00000000	*Section 47.2.3.46*
0x00FC–0x015F	Reserved			
0x0160–0x01FF	TSU Extension registers	Refer to *Chapter 49: Timestamping unit (TSU)*		

Table 498. M_CAN memory map (continued)

Address Offset	Register Name	Access(1)	Reset value	Location
0x0200–0x05FF	DMU (DMU registers start at 0x03C0)	Refer to *Chapter 48: DMA interface unit (DMU)*		
0x0600–0x0FFF	Reserved			

1. R = Read, S = Set on read, W = Write, P = Protected Write, C = Clear/preset on write, p = Protected set, X= Reset on read

47.2.3 Registers

47.2.3.1 Core release register (CREL)

Core release for coding of revisions (the coding of revisions depends on the module version used in the device).

Offset: 0x0000 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TS_MON								DAY							
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. Refer to *Table 498: M_CAN memory map* for reset value of this register.

Figure 754. Core Release Register (CREL)

Table 499. CREL field descriptions

Field	Description
31:28 REL	Core Release One digit, BCD.
27:24 STEP	Step of Core Release One digit, BCD.
23:20 SUBSTEP	Substep of Core Release One digit, BCD.
19:16 YEAR	Time Stamp Year One digit, BCD.
15:8 TS_MON	Time Stamp Month Two digits, BCD.
7:0 DAY	Time Stamp Day Two digits, BCD.

47.2.3.2 Endian register (ENDN)

Offset: 0x0004 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ETV															
W																
Reset	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ETV															
W																
Reset	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1

Figure 755. Endian Register (ENDN)

Table 500. ENDN field descriptions

Field	Description
31:0 ETV	Endianness Test Value The endianness test value is 0x87654321.

47.2.3.3 Data bit timing and prescaler register (DBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programmed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 M_CAN clock periods. tq = (DBRP + 1) M_CAN clock period.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. **DTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) **[DTSEG1 + DTSEG2 + 3]** tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning that the data for the next bit is available at the first clock edge after the sample point.

Offset: 0x000C Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TDC(1)	0	0	DBRP(1)				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	DTSEG1(1)					DTSEG2(1)				DSJW(1)			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 756. Data Bit Timing and Prescaler Register (DBTP)

Table 501. DBTP field descriptions

Field	Description
23 TDC	Transceiver Delay Compensation 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled
20:16 DBRP	Data Baud Rate Prescaler (0x00–0x1F) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. When TDC = 1, the range is limited to 0,1. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
12:8 DTSEG1	Data time segment before sample point (0x0–0x1F) Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
7:4 DTSEG2	Data time segment after sample point (0x0–0xF) Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
3:0 DSJW	Data (Re) Synchronization Jump Width (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a fast bit rate of 500 Kbit/s.

The bit rate configured for the CAN FD data phase via DBTP must be higher than or equal to the bit rate configured for the arbitration phase via NBTP.

To achieve high bit rates in CAN FD mode for the data phase up to 8 Mbit/s the number of time quanta bits per CAN bit can be chosen down to 4 time quanta per bit time.

47.2.3.4 Test register (TEST)

Write access to the Test Register has to be enabled by setting CCCR[TEST] to ‘1’. All Test Register functions are set to their reset values when CCCR[TEST] is reset. Loop Back mode and software control of M_CAN Tx pin are hardware test modes. Programming of TX ≠ ‘00’ may disturb the message transfer on the CAN bus.

Offset: 0x0010 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	SVAL	TXBNS				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PVAL	TXBNP					RX	TX(1)		LBCK(1)	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	—	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 757. Test Register (TEST)

Table 502. TEST field descriptions

Field	Description
21 SVAL	Started Valid 0 Value of TXBNS not valid 1 Value of TXBNS valid
20:16 TXBNS	Tx Buffer Number Started (0x00–0x1F) Tx Buffer number of message whose transmission was started last. Valid when SVAL is set. Valid values are 0 to 31.
13 PVAL	Prepared Valid 0 Value of TXBNP not valid 1 Value of TXBNP valid
12:8 TXBNP	Tx Buffer Number Prepared (0x00-0x1F) Tx Buffer number of message that is ready for transmission. Valid when PVAL is set. Valid values are 0 to 31.
7 RX	Receive Pin Monitors the actual value of CAN Rx pin. 0 The CAN bus is dominant (CAN Rx input = ‘0’) 1 The CAN bus is recessive (CAN Rx input = ‘1’)
6:5 TX	Control of Transmit Pin 00 Reset value, CAN Tx output is controlled by the CAN core, updated at the end of the CAN bit time. 01 Sample Point can be monitored at pin CAN Tx output 10 Dominant (‘0’) level at pin CAN Tx output 11 Recessive (‘1’) at pin CAN Tx output
4 LBCK	Loop Back mode 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (refer to *Section 47.2.5.9: Test modes*)

47.2.3.5 RAM watchdog register (RWD)

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access via the M_CAN's Generic manager interface starts the Message RAM Watchdog Counter with the value configured by the WDC bits. The counter is reloaded with WDC bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag WDI bit of the IR is set. The RAM Watchdog Counter is clocked by the Host clock.

Offset: 0x0014 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WDV								WDC(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 758. RAM Watchdog Register (RWD)

Table 503. RWD field descriptions

Field	Description
15:8 WDV	Watchdog Value Actual Message RAM Watchdog Counter Value.
7:0 WDC	Watchdog Configuration Start value of the Message RAM Watchdog Counter. With the reset value of '00' the counter is disabled.

47.2.3.6 CC control register (CCCR)

The following section shows the CAN Control register.For details about setting and resetting of single bits refer to *Section 47.2.5.1.1: Software initialization*.

Offset: 0x0018 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NISO(1)	TXP(1)	EFBI(1)	PXHD(1)	WMM(1)	UTSU(1)	BRSE(1)	FDOE(1)	TEST_EN(2)	DAR(1)	MON(2)	CSR	CSA	ASM(2)	CCE(1)	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.
2. These are protected set (p) bits.

Figure 759. CC Control register (CCCR)

Table 504. CCCR field descriptions

Field	Description
15 NISO	Non ISO Operation If this bit is set, the M_CAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. 0 CAN FD frame format according to ISO11898-1 1 CAN FD frame format according to Bosch CAN FD Specification V1.0
14 TXP	Transmit Pause If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame. 0 Transmit pause disabled 1 Transmit pause enabled
13 EFBI	Edge Filtering during Bus Integration 0 Edge filtering disabled 1 Two consecutive dominant tq required to detect an edge for hard synchronization
12 PXHD	Protocol Exception Handling Disable 0 Protocol exception handling enabled 1 Protocol exception handling disabled **Note:** When protocol exception handling is disabled, the M_CAN transmits an error frame when it detects a protocol exception condition.
11 WMM	Wide Message Marker Enables the use of 16-bit Wide Message Markers. When 16-bit Wide Message Markers are used (WMM = 1), 16-bit internal timestamping is disabled for the Tx Event FIFO. 0 8-bit Message Marker used 1 16-bit Message Marker used, replacing 16-bit timestamps in Tx Event FIFO

Table 504. CCCR field descriptions (continued)

Field	Description
10 UTSU	Use Timestamping Unit When UTSU is set, 16-bit Wide Message Markers are also enabled regardless of the value of WMM. 0 Internal time stamping 1 External time stamping by TSU **Note:** When generic parameter connected_tsu_g = 0, there is no TSU connected to the M_CAN. In this case bit UTSU is fixed to zero by synthesis.
9 BRSE	Bit Rate Switch Enable 0 Bit rate switching for transmissions disabled 1 Bit rate switching for transmissions enabled **Note:** When CAN FD operation is disabled FDOE = 0, BRSE is not evaluated.
8 FDOE	FD Operation Enable 0 FD operation disabled 1 FD operation enabled
7 TEST_EN	Test Mode Enable 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
6 DAR	Disable Automatic Retransmission 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
5 MON	Bus Monitoring Mode Bit MON can only be set by the Host when both CCE and INIT are set to 1. The bit can be reset by the Host at any time. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
4 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA are set after all pending transfer requests have been completed and the CAN bus reached idle.
3 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_CAN may be set in power down by stopping Host clock and core clock
2 ASM	Restricted Operation Mode Bit ASM is only set by the Host when both CCE and INIT are set to 1. The bit is reset by the Host at any time. 0 Normal CAN operation 1 Restricted Operation Mode active

Table 504. CCCR field descriptions (continued)

Field	Description
1 CCE	Configuration Change Enable 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR.INIT = 1)
0 INIT	Initialization 0 Normal Operation 1 Initialization is started **Note:** Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

47.2.3.7 Nominal bit timing and prescaler register (NBTP)

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programmed in the range of [5…385] time quanta. The CAN time quantum may be programmed in the range of [1…512] M_CAN clock periods.

Equation 22 tq = (NBRP + 1) M_CAN clock period

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Offset: 0x001C Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NSJW(1)								NBRP(1)							
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NTSEG1(1)								0	NTSEG2(1)						
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 760. Nominal Bit Timing and Prescaler Register (NBTP)

Table 505. NBTP field descriptions

Field	Description
31:25 NSJW	(Re) Synchronization Jump Width (0x00–0x7F) Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
24:16 NBRP	Baud Rate Prescaler (0x000–0x1FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
15:8 NTSEG1	Time segment before sample point (0x01–0xFF) Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
6:0 NTSEG2	The time segment after the sample point (0x01–0x7F) Valid values are 1 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x06000A03 configures the M_CAN for a bit rate of 500 Kbit/s.

47.2.3.8 Timestamp counter configuration register (TSCC)

Configuration of the internal 16-bit Timestamp Counter. The use of the internal Timestamp Counter is enabled by CCCR.UTSU = 0. Handling of internal/external timestamps is described in *Section 47.2.6: Timestamp generation*.

Offset: 0x0020 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	TCP(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TSS(1)	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 761. Timestamp Counter Configuration Register (TSCC)

Table 506. TSCC field descriptions

Field	Description
19:16 TCP	Timestamp Counter Prescaler (0x0–0xF) Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. **Note:** with CAN FD an external counter is required for timestamp generation (TSS = 10).
1:0 TSS	Timestamp Select 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to Timestamp Counter Prescaler (TCP) 10 External timestamp counter value used 11 Same as 00 **Note:** in case of CAN FD mode an external counter (available through PIT/GST) is requested for timestamp generation.

47.2.3.9 Timestamp counter value register (TSCV)

Offset: 0x0024 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TSC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 762. Timestamp Counter Value Register (TSCV)

Table 507. TSCV field descriptions

Field	Description
15:0 TSC	Timestamp Counter The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC.TSS = ‘01’, the Timestamp Counter is incremented in multiples of CAN bit times [1…16] depending on the configuration of TSCC.TCP. A wrap around sets interrupt flag IR.TSW. Write access resets the counter to zero. When TSCC.TSS = ‘10’, TSC reflects the external Timestamp Counter value. A write access has no impact.

Note: A "wrap around" is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

Byte access: writing one of the register bytes 3/2/1/0 resets the Timestamp Counter.

47.2.3.10 Timeout counter configuration register (TOCC)

For a description of the Timeout Counter, refer to *Section 47.2.7*.

Offset: 0x0028 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TOP(1)															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TOS(1)		ETOC(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 763. Timeout Counter Configuration Register (TOCC)

Table 508. TOCC field descriptions

Field	Description
31:16 TOP	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
2:1 TOS	Timeout Select When operating in Continuous mode, a write to **TOCV** presets the counter to the value configured by **TOCC[TOP]** and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by **TOCC[TOP]**. Down-counting is started when the first FIFO element is stored. 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
0 ETOC	Enable Timeout Counter 0 Timeout Counter disabled 1 Timeout Counter enabled **Note:** For use of timeout function with CAN FD refer to *Section 47.2.7: Timeout counter*.

47.2.3.11 Timeout counter value register (TOCV)

Offset: 0x002C Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TOC															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 764. Timeout Counter Value Register (TOCV)

Table 509. TOCV field descriptions

Field	Description
15:0 TOC	The Timeout Counter is decremented in multiples of CAN bit times (1 to 16) depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

Note: Byte access: when TOCC.TOS = 0, writing one of the register bytes 3/2/1/0 resets the Timeout Counter.

47.2.3.12 Error counter register (ECR)

Offset: 0x0040 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	CEL(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RP	REC							TEC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Access type is RX: reset on read.

Figure 765. Error Counter Register (ECR)

Table 510. ECR field descriptions

Field	Description
23:16 CEL	CAN Error Logging The counter is incremented each time when a CAN protocol error causes the 8-bit Transmit Error Counter TEC or the 7-bit Receive Error Counter REC to be incremented. The counter is also incremented when the Bus_Off limit is reached. It is not incremented when only RP is set without changing REC. The increment of CEL follows after the increment of REC or TEC. The counter is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR.ELO. **Note:** Byte access: Reading byte 2 resets CEL to zero, reading bytes 3/1/0 has no impact.
15 RP	Receive Error Passive 0 The Receive Error Counter is below the error passive level of 128 1 The Receive Error Counter has reached the error passive level of 128
14:8 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127.
7:0 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255.

Note: When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

47.2.3.13 Protocol status register (PSR)

Offset: 0x0044 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	TDCV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	PXE(1)	RFDF(1)	RBRS(1)	RESI(1)	DLEC(2)			BO	EW	EP	ACT		LEC(2)		
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

1. Access type is RX: reset on read.
2. Access type is RS: set on read.

Figure 766. Protocol Status Register (PSR)

Table 511. PSR field descriptions

Field	Description
22:16 TDCV	Transmitter Delay Compensation Value (0x00-0x7F) Position of the secondary sample point, defined by the sum of the measured delay from CAN Tx to CAN Rx and TDCR.TDCO. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.
14 PXE	Protocol Exception Event 0 No protocol exception event occurred since last read access 1 Protocol exception event occurred **Note:** RBRS, RESI and DLEC.
13 RFDF	Received CAN FD Message This bit is set independent of acceptance filtering. 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with FDF flag set has been received **Note:** Byte access: reading byte 0 resets RFDF, reading bytes 3/2/1 has no impact.
12 RBRS	BRS flag of last received CAN FD Message This bit is set together with RFDF, independent of acceptance filtering. 0 Last received CAN FD message did not have its BRS flag set. 1 Last received CAN FD message had its BRS flag set. **Note:** Byte access: reading byte 0 resets RBRS, reading bytes 3/2/1 has no impact.
11 RESI	ESI CAN FD Message with ESI flag This bit is set together with RFDF, independent of acceptance filtering. 0 Last received CAN FD message did not have its ESI flag set. 1 Last received CAN FD message had its ESI flag set **Note:** Byte access: reading byte 0 resets RESI, reading bytes 3/2/1 has no impact.
10:8 DLEC	Data Phase Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field is cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. **Note:** Byte access: reading byte 0 sets DLEC to 0x7, reading bytes 3/2/1 has no impact.
7 BO	Bus_Off Status 0 The M_CAN is not Bus_Off 1 The M_CAN is in Bus_Off state
6 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
5 EP	Error Passive 0 The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_CAN is in the Error_Passive state

Table 511. PSR field descriptions (continued)

Field	Description
4:3 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing – node is synchronizing on CAN communication 01 Idle-node is neither receiver nor transmitter 10 Receiver-node is operating as receiver 11 Transmitter-node is operating as transmitter
2:0 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field is cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 Form Error: A fixed format part of a received frame has the wrong format. 011 AckError: The message transmitted by the M_CAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register. **Note:** When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) is shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence is shown as a Form Error, not Stuff Error. **Note:** The Bus_Off recovery sequence (refer to CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT.] If the device goes Bus_Off, it sets CCCR[INIT] of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 ´ 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters is resets. During the wag time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit 0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences. **Note:** Byte access: reading byte 0 sets LEC to 0x7, reading bytes 3/2/1 has no impact.

47.2.3.14 Transmitter delay compensation register (TDCR)

Offset: 0x0048 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TDCO(1)							0	TDCF(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 767. Transmitter Delay compensation Register (TDCR)

Table 512. TDCR field descriptions

Field	Description
14:8 TDCO	Transmitter Delay Compensation SSP Offset (0x00-0x7F) Offset value defining the distance between the measured delay from CAN Tx to CAN Rx and the secondary sample point. Valid values are 0 to 127 mtq.
6:0 TDCF	Transmitter Delay Compensation Filter Window Length (0x00-0x7F) Defines the minimum value for the SSP position, dominant edges on CAN Rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq.

47.2.3.15 Interrupt register (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset clears the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Offset: 0x0050 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 768. Interrupt Register (IR)

Table 513. IR field descriptions

Field	Description
29 ARA	Access to Reserved Address 0 No access to reserved address occurred 1 Access to reserved address occurred
28 PED	Protocol Error in Data Phase (Data Bit Time is used) 0 No protocol error in data phase 1 Protocol error in data phase detected (PSR.DLEC ≠ 0,7)
27 PEA	Protocol Error in Arbitration Phase (Nominal Bit Time is used) 0= No protocol error in arbitration phase 1= Protocol error in arbitration phase detected (PSR.LEC ≠ 0,7)
26 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
25 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed
24 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed
23 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
22 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred

Table 513. IR field descriptions (continued)

Field	Description
21 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to '1'. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (for example parity logic)
20 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (for example ECC)
19 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
18 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
17 MRAF	Message RAM Access Failure The flag is set, when the Rx Handler – has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. – was not able to write a message to the Message RAM. In this case message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler is not able to read a message from the Message RAM in time. In this case message transmission is aborted. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM. 0 No Message RAM access failure occurred 1 Message RAM access failure occurred
16 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
15 TEFL	Tx Event FIFO Event Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero

Table 513. IR field descriptions (continued)

Field	Description
14 TEFF	Tx Event FIFO Full
13 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark
12 TEFN	Tx Event FIDO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
11 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
10 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
9 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
8 HPM	High Priority Message 0 No high priority message received 1 High priority message received
7 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
6 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
5 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
4 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
3 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
2 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full

Table 513. IR field descriptions (continued)

Field	Description
1 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
0 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

47.2.3.16 Interrupt enable register (IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt register is signaled on an interrupt line.

Offset: 0x0054 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 769. Interrupt Enable Register (IE)

Table 514. IE field descriptions

Field	Description
29 ARAE	Access to Reserved Address Enable 0 Interrupt disabled 1 Interrupt enabled
28 PEDE	Protocol Error in Data Phase Enable 0 Interrupt disabled 1 Interrupt enabled
27 PEAE	Protocol Error in Arbitration Phase Enable 0 Interrupt disabled 1 Interrupt enabled
26 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 514. IE field descriptions (continued)

Field	Description
24 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
21 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 MRAFE	Message RAM Access Failure Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
13 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 514. IE field descriptions (continued)

Field	Description
10 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
5 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
4 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
3 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
2 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
1 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
0 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

47.2.3.17 Interrupt line select register (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

Offset: 0x0058 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 770. Interrupt Line Select Register (ILS)

Table 515. ILS field descriptions

Field	Description
29 ARAL	Access to Reserved Address Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
28 PEDL	Protocol Error in Data Phase Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
27 PEAL	Protocol Error in Arbitration Phase Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
26 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
25 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
24 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
23 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
22 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
21 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 515. ILS field descriptions (continued)

Field	Description
20 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
19 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
18 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
17 MRAFL	Message RAM Access Failure Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
16 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
15 TEFLL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
14 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
13 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
12 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
11 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
10 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
9 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
8 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
7 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 515. ILS field descriptions (continued)

Field	Description
6 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
5 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
4 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
3 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
2 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
1 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
0 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

47.2.3.18 Interrupt line enable register (ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

Offset: 0x005C Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EINT1	EINT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 771. Interrupt Line Enable Register (ILE)

Table 516. ILE field descriptions

Field	Description
1 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled
0 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 Interrupt line enabled

47.2.3.19 Global filter configuration register (GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in *Figure 810* and *Figure 811*.

Offset: 0x0080 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	ANFS(1)		ANFE(1)		RRFS(1)	RRFE(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 772. Global Filter Configuration Register (GFC)

Table 517. GFC field descriptions

Field	Description
5:4 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
3:2 ANFE	ANFE[1:0]: Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject

Table 517. GFC field descriptions (continued)

Field	Description
1 RRFS	Reject Remote Frames Standard 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
0 RRFE	Reject Remote Frames Extended 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

47.2.3.20 Standard ID filter configuration register (SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in *Figure 804*.

Offset: 0x0084 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	LSS(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLSSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 773. Standard ID Filter Configuration register (SIDFC)

Table 518. SIDFC field descriptions

Field	Description
23:16 LSS	List Size Standard 0x00: No standard Message ID filter 0x01: Number of standard Message ID filter elements ... 0x80: Number of standard Message ID filter elements 0x81: Values greater than 128 are interpreted as 128 ... 0xFF: Values greater than 128 are interpreted as 128
15:2 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, refer to *Figure 800*).

47.2.3.21 Extended ID filter configuration register (XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in *Figure 805*.

Offset: 0x0088 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	LSE(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLESA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 774. Extended ID Filter Configuration register (XIDFC)

Table 519. XIDFC field descriptions

Field	Description
22:16 LSE	List Size Extended 0x00: No extended Message ID filter 0x01: Number of extended Message ID filter elements ... 0x40: Number of extended Message ID filter elements 0x41: Values greater than 64 are interpreted as 64 ... 0x3F: Values greater than 64 are interpreted as 64
15:2 FLESA	Filter List Extended Start Address Start address of extended Message ID filter list (32-bit word address, refer to *Figure 800*).

47.2.3.22 Extended ID and mask register (XIDAM)

Offset: 0x0090 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	EIDM(1)												
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EIDM(1)															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 775. Extended ID and Mask Register (XIDAM)

Table 520. XIDAM field descriptions

Field	Description
28:0 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

47.2.3.23 High priority message status register (HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Offset: 0x0094 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLST	FIDX							MSI		BIDX					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 776. High Priority Message Status Register (HPMS)

Table 521. HPMS field descriptions

Field	Description
15 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
14:8 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
7:6 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
5:0 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = ‘1’.

47.2.3.24 New data 1 register (NDAT1)

Offset: 0x0098 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 777. New Data 1 Register (NDAT1)

Table 522. NDAT1 field descriptions

Field	Description
31:0 ND[31:0]	New Data[31:0] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset clears the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

47.2.3.25 New data 2 register (NDAT2)

Offset: 0x009C Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 778. New Data 2 Register (NDAT2)

Table 523. NDAT2 field descriptions

Field	Description
31:0 ND[63:32]	New Data[63:32] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset clears the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

47.2.3.26 Rx FIFO 0 configuration register (RXF0C)

Offset: 0x00A0 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	F0OM	F0WM(1)							0	F0S(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	F0SA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 779. Rx FIFO 0 Configuration Register (RXF0C)

Table 524. RXF0C field descriptions

Field	Description
31 F0OM	FIFO 0 Operation Mode FIFO 0 can be operated in blocking or in overwrite mode. 0 FIFO 0 blocking mode 1 FIFO 0 overwrite mode
30:24 F0WM	Rx FIFO 0 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) 0x40: Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) 0x41: Watermark interrupt disabled 0x3F: Watermark interrupt disabled

Table 524. RXF0C field descriptions (continued)

Field	Description
22:16 F0S	Rx FIFO 0 Size 0x00: No Rx FIFO 0 0x01: Number of Rx FIFO 0 elements ... 0x40: Number of Rx FIFO 0 elements 0x41: Values greater than 64 are interpreted as 64 ... 0x3F: Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to F0S-1.
15:2 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, *Figure 800*).

47.2.3.27 Rx FIFO 0 status register (RXF0S)

Offset: 0x00A4 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	RF0L	F0F	0	0	F0PI					
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	F0GI						0	F0FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 780. Rx FIFO 0 Status Register (RXF0S)

Table 525. RXF0S field descriptions

Field	Description
25 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero **Note:** Overwriting the oldest message when RXF0C.F0OM = '1' does not set this flag.
24 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
21:16 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63.

Table 525. RXF0S field descriptions (continued)

Field	Description
13:8F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.
6:0 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

47.2.3.28 Rx FIFO 0 acknowledge register (RXF0A)

Offset: 0x00A8 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	F0AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 781. Rx FIFO 0 Acknowledge Register (RXF0A)

Table 526. RXF0A field descriptions

Field	Description
5:0 F0AI	Rx FIFO 0 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This sets the Rx FIFO 0 Get Index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL].

47.2.3.29 Rx buffer configuration register (RXBC)

Offset: 0x00AC Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RBSA[1]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 782. Rx Buffer Configuration Register (RXBC)

Table 527. RXBC field descriptions

Field	Description
15:2 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C.

47.2.3.30 Rx FIFO 1 configuration register (RXF1C)

Offset: 0x00B0 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	F1OM[1]	F1WM[1]							0	F1S[1]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	F1SA[1]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 783. Rx FIFO 1 Configuration Register (RXF1C)

Table 528. RXF1C field descriptions

Field	Description
31 F1OM	FIFO 1 Operation Mode FIFO 1 can be operated in blocking or in overwrite mode. 0 FIFO 1 blocking mode 1 FIFO 1 overwrite mode
30:24 F1WM	Rx FIFO 1 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 1 watermark interrupt IR[RF1W] ... 0x40: Level for Rx FIFO 1 watermark interrupt IR[RF1W] 0x41: Watermark interrupt disabled ... 0x7F: Watermark interrupt disabled
22:16 F1S	F1S[6:0]: Rx FIFO 1 Size 0x00: No Rx FIFO 1 0x01: Number of Rx FIFO 1 elements ... 0x40: Number of Rx FIFO 1 elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to F1S - 1
15:2 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, refer to *Figure 800*).

47.2.3.31 Rx FIFO 1 status register (RXF1S)

Offset: 0x00B4 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMS		0	0	0	0	RF1L	F1F	0	0	F1PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	F1GI						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 784. Rx FIFO 1 Status Register (RXF1S)

Table 529. RXF1S field descriptions

Field	Description
31:30 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
25 RF1L	Rx FIFO 1 Message Lost This bit is a copy of interrupt flag IR[RF1L]. When IR[RF1L] is reset, this bit is also reset. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero **Note:** Overwriting the oldest message when RXF1C.F1OM = '1' does not set this flag.
24 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
21:16 F1PI	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
13:8 F1GI	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
6:0 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

47.2.3.32 Rx FIFO 1 acknowledge Register (RXF1A)

Offset: 0x00B8 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	F1AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 785. Rx FIFO 1 Acknowledge Register (RXF1A)

Table 530. RXF1A field descriptions

Field	Description
5:0 F1AI	Rx FIFO 1 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This sets the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].

47.2.3.33 Rx Buffer / FIFO Element Size Configuration (RXESC)

It configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

Offset: 0x00BC Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	RBDS(1)				F1DS(1)				F0DS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 786. Rx Buffer / FIFO Element Size Configuration Register (RXESC)

Table 531. RXESC field descriptions

Field	Description
10:8 RBDS	Rx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
6:4 F1DS	Rx FIFO 1 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
2:0 F0DS	Rx FIFO 0 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

Note: In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by RXESC is stored to the Rx Buffer respectively, Rx FIFO element. The rest of the frame's data field is ignored.

47.2.3.34 Tx Buffer Configuration Register (TXBC)

Offset: 0x00C0 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	TFQM(1)	TFQS(1)						0	0	NDTB(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 787. Tx Buffer Configuration Register (TXBC)

Table 532. TXBC field descriptions

Field	Description
30 TFQM	Tx FIFO/Queue Mode 0 Tx FIFO operation 1 Tx Queue operation
29:24 TFQS	Tx FIFO/Queue Size 0x00: No Tx FIFO/Queue 0x01: Number of Tx Buffers used for Tx FIFO/Queue ... 0x20: Number of Tx Buffers used for Tx FIFO/Queue 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
21:16 NDTB	Number of Dedicated Transmit Buffers 0x00: No Dedicated Tx Buffers 0x01: Number of Dedicated Tx Buffers ... 0x20: Number of Dedicated Tx Buffers 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
15:2 TBSA	Tx Buffers Start Address Start address of Tx Buffers section in Message RAM (32-bit word address, refer to *Figure 800*).

Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

47.2.3.35 Tx FIFO/Queue Status Register (TXFQS)

Offset: 0x00C4 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	TFQF	TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	TFGI					0	0	TFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 788. Tx FIFO/Queue Status Register (TXFQS)

Table 533. TXFQS field descriptions

Field	Description
21 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
20:16 TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31.
12:8 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = ‘1’).
5:0 TFFL	Tx FIFO Free Level Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = ‘1’).

Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get indexes indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

47.2.3.36 Tx Buffer Element Size Configuration (TXESC)

It configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

Offset: 0x00C8 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TBDS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 789. Tx Buffer Element Size Configuration Register (TXESC)

Table 534. TXESC field descriptions

Field	Description
2:0 TBDS	Tx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field **Note:** In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as ‘0xCC’ (padding bytes).

47.2.3.37 Tx Buffer Request Pending Register (TXBRP)

Offset: 0x00CC Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 790. Tx Buffer Request Pending Register (TXBRP)

Table 535. TXBRP field descriptions

Field	Description
31:0 TRP[31:0]	Transmission Request Pending Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan (refer to *Section 47.2.11: Tx handling*) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via TXBCF – after successful transmission together with the corresponding TXBTO bit – when the transmission has not yet been started at the point of cancellation – when the transmission has been aborted due to lost arbitration – when an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

Note: TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.

47.2.3.38 Tx Buffer Add Request Register (TXBAR)

Offset: 0x00D0 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 791. Tx Buffer Add Request Register (TXBAR)

Table 536. TXBAR field descriptions

Field	Description
31:0 AR[31:0]	Add Request Each Tx Buffer has its own Add Request bit. Writing a ‘1’ sets the corresponding Add Request bit; writing a ‘0’ has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

Note: If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

47.2.3.39 Tx Buffer Cancellation Request Register (TXBCR)

Offset: 0x00D4 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 792. Tx Buffer Cancellation Request Register (TXBCR)

Table 537. TXBCR field descriptions

Field	Description
31:0 CR[31:0]	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing a ‘1’ sets the corresponding Cancellation Request bit; writing a ‘0’ has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

47.2.3.40 Tx Buffer Transmission Occurred Register (TXBTO)

Offset: 0x00D8 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 793. Tx Buffer Transmission Occurred Register (TXBTO)

Table 538. TXBTO field descriptions

Field	Description
31:0 TO[31:0]	Transmission Occurred Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register TXBAR. 0 No transmission occurred 1 Transmission occurred

47.2.3.41 Tx Buffer Cancellation Finished Register (TXBCF)

Offset: 0x00DC Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 794. Tx Buffer Cancellation Finished Register (TXBCF)

Table 539. TXBCF field descriptions

Field	Description
31:0 CF[31:0]	Cancellation Finished Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

47.2.3.42 Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Offset: 0x00E0 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 795. Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Table 540. TXBTIE field descriptions

Field	Description
31:0 TIE[31:0]	Transmission Interrupt Enable Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

47.2.3.43 Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Offset: 0x00E4 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 796. Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Table 541. TXBCIE field description

Field	Description
31:0 CFIE[31:0]	Cancellation Finished Interrupt Enable Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

47.2.3.44 Tx Event FIFO Configuration Register (TXEFC)

Offset: 0x00F0 Access: Read/Write Protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		EFWM(1)						0		EFS(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EFSA(1)														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 797. Tx Event FIFO Configuration Register (TXEFC)

Table 542. TXEFC field descriptions

Field	Description
29:24 EFWM	Event FIFO Watermark 0x00: Watermark interrupt disabled 0x01: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) ... 0x20: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) 0x21: Watermark interrupt disabled ... 0x3F: Watermark interrupt disabled
21:16 EFS	Event FIFO Size 0x00: Tx Event FIFO disabled 0x01: Number of Tx Event FIFO elements ... 0x20: Number of Tx Event FIFO elements 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to EFS - 1
15:2 EFSA	Event FIFO Start Address. Start address of Tx Event FIFO in Message RAM (32-bit word address,*Figure 800*.)

47.2.3.45 Tx Event FIFO Status Register (TXEFS)

Offset: 0x00F4 Access: Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	TEFL	EFF	0	0	0	EFPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	EFGI					0	0	EFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 798. Tx Event FIFO Status Register (TXEFS)

Table 543. TXEFS field descriptions

Field	Description
25 TEFL	Tx Event FIFO Element Lost This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset. 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
24 EFF	Event FIFO Full 0 Tx Event FIFO not full 1 Tx Event FIFO full
20:16 EFPI	Event FIFO Put Index Tx Event FIFO write index pointer, range 0 to 31.
12:8 EFGI	Event FIFO Get Index Tx Event FIFO read index pointer, range 0 to 31.
5:0 EFFL	Event FIFO Fill Level Number of elements stored in Tx Event FIFO, range 0 to 32.

47.2.3.46 Tx Event FIFO Acknowledge Register (TXEFA)

Offset: 0x00F8 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	EFAI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 799. Tx Event FIFO Acknowledge Register (TXEFA)

Table 544. TXEFA field descriptions

Field	Description
4:0 EFAI	Event FIFO Acknowledge Index After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This sets the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the FIFO 0 Fill Level TXEFS[EFFL].

47.2.4 Message RAM

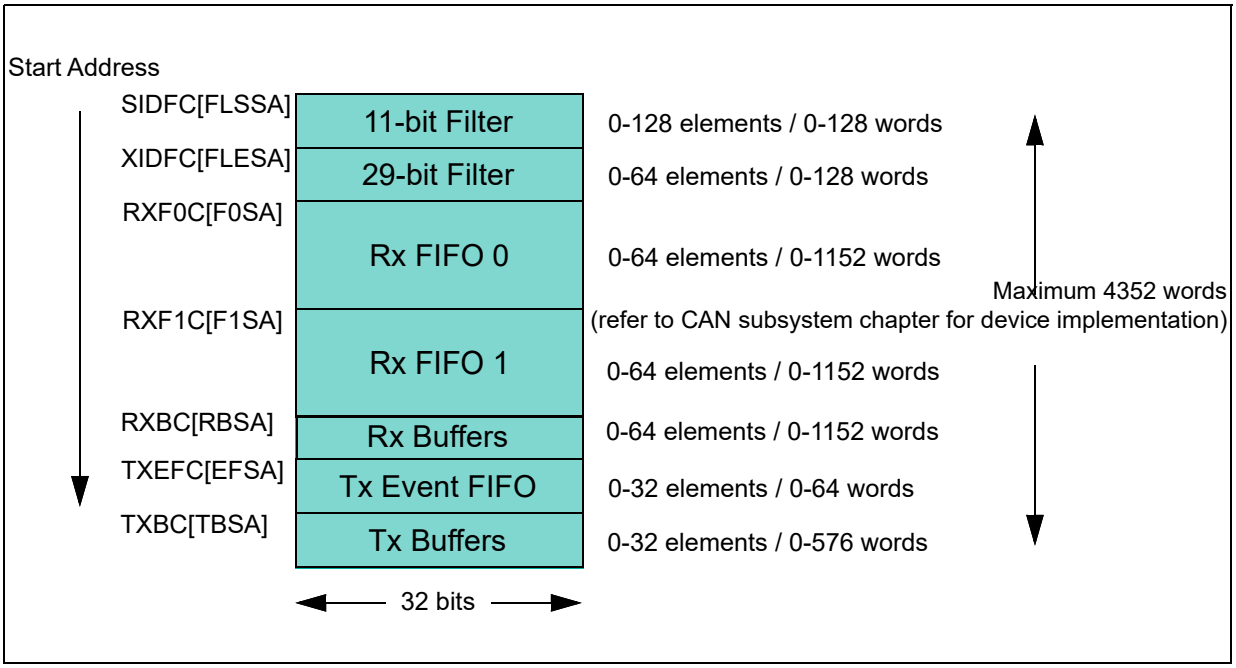
For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M_CAN module.

Note: Because the Message RAM is equipped with ECC functionality, it is recommended to initialize the Message RAM after hardware reset by writing, for example, 0x00000000 to each Message RAM word to create valid parity/ECC checksums. This avoids that reading from uninitialized Message RAM sections activates interrupt IR.BEC (Bit Error Corrected) or IR.BEU (Bit Error Uncorrected).

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_CAN module can be configured to allocate up to 4352 words in the Message RAM (for the chip-specific implementation details of each module's instance, refer to device configuration chapter). It is not necessary to configure each of the sections listed in *Figure 800*, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via RXESC.F0DS, RXESC.F1DS, RXESC.RBDS, and TXESC.TBDS.

Figure 800. Message RAM configuration



When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses such as only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

47.2.4.1 Rx buffer and FIFO element

Up to 64 Rx buffers and two Rx FIFOs can be configured in the message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx buffer / FIFO element is shown in the following figure. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESC.

R1A: When no TSU is used (CCCR.UTSU = 0), R1A.RXTS[15:0] holds the 16-bit timestamp generated by the M_CAN's internal timestamping logic.

R1B: When a TSU is used (CCCR.UTSU = 1) and when bit SSYNC/ESYNC of the matching filter element is set, R1B.TSC = 1 and R1B.RXTSP[3:0] holds the number of the TSU's Timestamp register which holds the 32-bit timestamp captured by the TSU. Else R1B.TSC = 0 and R1B.RXTSP[3:0] is not valid.

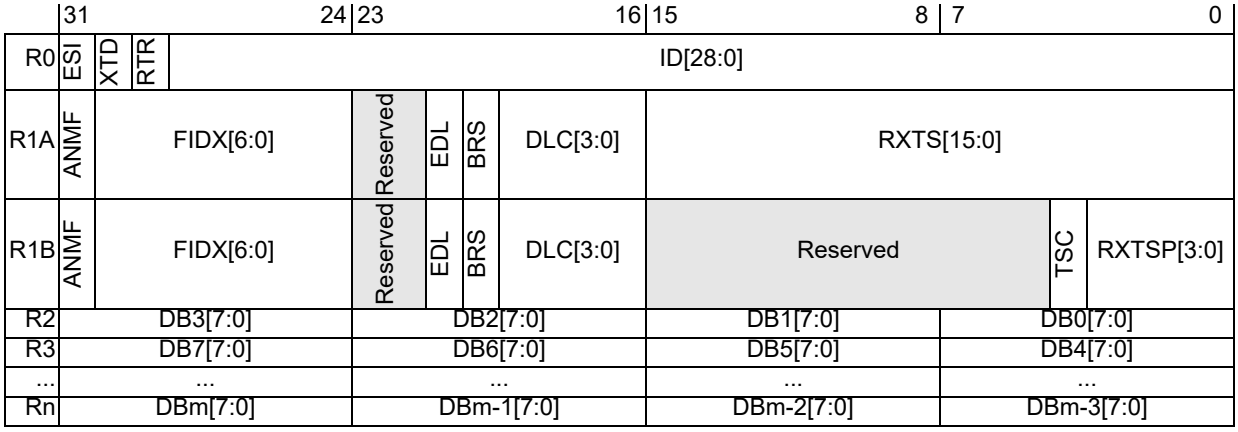


Figure 801. Rx buffer and FIFO element

Table 545. Rx buffer and FIFO element descriptions

Field	Description
R0 bit 31 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 bit 30 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier. 0 11-bit standard identifier 1 29-bit extended identifier
R0 bit 29 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame **Note:** There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = '1'), bit RTR reflects the state of the reserved bit r1.
R0 bits 28:0 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1A/B bit 31 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE]. 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element
R1A/B bits 30:24 FIDX[6:0]: Filter Index	0-127=Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.

Table 545. Rx buffer and FIFO element descriptions (continued)

Field	Description
R1A/B bit 21 EDL: Extended Data Length	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
R1A/B bit 20 BRS: Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1A/B bits 19:16 DLC[3:0]: Data Length Code	0-8 CAN + CAN FD: Received frame has 0-8 data bytes 9-15 CAN: Received frame has 8 data bytes 9-15 CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1A bits 15:0 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R1B bit 4 TSC: Timestamp Captured	0 No timestamp captured 1 Timestamp captured and stored in TSU Timestamp register referenced by R1B.RXTSP
R1B bits 3:0 RXTSP[3:0]: Rx Timestamp Pointer	Number of TSU Time Stamp register (TS0–15) where the related timestamp is stored.
R2 bits 31:24	DB3[7:0]: Data Byte 3
R2 bits 23:16	DB2[7:0]: Data Byte 2
R2 bits 15:8	DB1[7:0]: Data Byte 1
R2 bits 7:0	DB0[7:0]: Data Byte 0
R3 bits 31:24	DB7[7:0]: Data Byte 7
R3 bits 23:16	DB6[7:0]: Data Byte 6
R3 bits 15:8	DB5[7:0]: Data Byte 5
R3 bits 7:0	DB4[7:0]: Data Byte 4
...	...
Rn bits 31:24	DBm[7:0]: Data Byte m
Rn bits 23:16	DBm-1[7:0]: Data Byte m-1
Rn bits 15:8	DBm-2[7:0]: Data Byte m-2
Rn bits 7:0	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words (Rn = 2 to 17) are used for storage of a CAN message's data field.

47.2.4.2 Tx buffer element

The Tx buffers section can be configured to hold dedicated Tx buffers as well as a Tx FIFO / Tx queue. In case that the Tx buffers section is shared by dedicated Tx buffers and a Tx

FIFO / Tx queue, the dedicated Tx buffers start at the beginning of the Tx buffers section followed by the buffers assigned to the Tx FIFO or Tx queue. The Tx handler distinguishes between dedicated Tx buffers and Tx FIFO / Tx queue by evaluating the Tx buffer configuration TXBC.TFQS and TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESC.

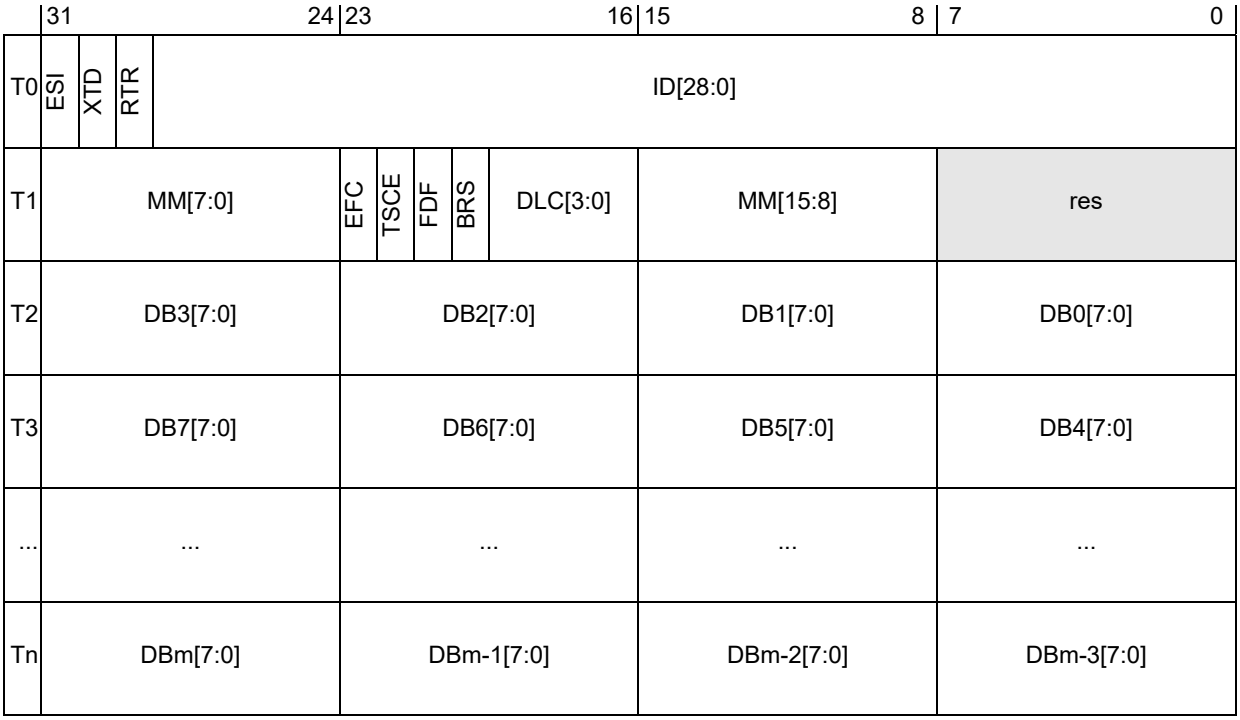


Figure 802. Tx buffer element

Table 546. Tx buffer element descriptions

Field	Description
T0 bit 31 ESI: Error State Indicator	0 ESI bit in CAN FD format depends only on error passive flag 1 ESI bit in CAN FD format transmitted recessive **Note:** The ESI bit of the transmit buffer is OR'ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node always transmit the ESI bit recessive
T0 bit 30 XTD: Extended Identifier	0 11-bit standard identifier 129-bit extended identifier
T0 bit 29 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame **Note:** When RTR = 1, the M_CAN transmits a remote frame according to ISO11898-1, even if CCCR.FDOE enables the transmission in CAN FD format.
T0 bits 28:0 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Table 546. Tx buffer element descriptions (continued)

Field	Description
T1 bits 31:24 MM[7:0]: Message Marker	Written by CPU during Tx buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
T1 bit 23 EFC Event FIFO Control	0 Do not store Tx events 1 Store Tx events
T1 bit 22 TSCE Time Stamp Capture Enable for TSU	Only available when CCCR.UTSU = '1'. When this bit is set and the message is transmitted, a pulse with the duration of one m_can_hclk period is generated at output m_can_tsrx to signal the transmission of a Sync message to the Timestamping Unit (TSU) connected to the M_CAN. 0 Timestamp capture disabled 1 Timestamp capture enabled
T1 bit 21 FDF FD Format	0 Frame transmitted in classic CAN format 1 Frame transmitted in CAN FD format
T1 bit 20 BRS Bit Rate Switch	0 CAN FD frames transmitted without bit rate switching 1 CAN FD frames transmitted with bit rate switching
T1 bits 19:16 DLC[3:0] Data Length Code	0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes 9-15 CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes
T1 bits 19:16 MM[15:8] Message Marker	High byte of wide message marker, written by CPU during Tx buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status. Available only when CCCR.WMM = 1 or when CCCR.UTSU = 1.
T2 bits 31:24	DB3[7:0]: Data Byte 3
T2 bits 23:16	DB2[7:0]: Data Byte 2
T2 bits 15:8	DB1[7:0]: Data Byte 1
T2 bits 7:0	DB0[7:0]: Data Byte 0
T3 bits 31:24	DB7[7:0]: Data Byte 7
T3 bits 23:16	DB6[7:0]: Data Byte 6
T3 bits 15:8	DB5[7:0]: Data Byte 5
T3 bits 7:0	DB4[7:0]: Data Byte 4
...	...
Tn bits 31:24	DBm[7:0]: Data Byte m
Tn bits 23:16	DBm-1[7:0]: Data Byte m-1
Tn bits 15:8	DBm-2[7:0]: Data Byte m-2
Tn bits 7:0	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 2 to 17) are used for storage of a CAN message's data field.

47.2.4.3 Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event FIFO the host CPU gets this information in the order the messages were transmitted. Status information about the Tx event FIFO can be obtained from register TXEFS.

E1A: When CCCR.WMM = 0 and no TSU is used (CCCR.UTSU = 0), E1A.TXTS[15:0] holds the 16-bit timestamp generated by the M_CAN's internal timestamping logic.

E1B: When 16-bit message markers are enabled (CCCR.WMM ='1') or when CCCR.UTSU = 1, E1B.MM[15:8] holds the upper 8 bit of the wide message marker. When a TSU is used (CCCR.UTSU = 1) and when bit TSCE of the related Tx buffer element is set, E1B.TSC = 1 and E1B.TXTSP[3:0] holds the number of the TSU's timestamp register which holds the 32-bit timestamp captured by the TSU. Else E1B.TSC = 0 and E1B.TXTSP[3:0] is not valid.

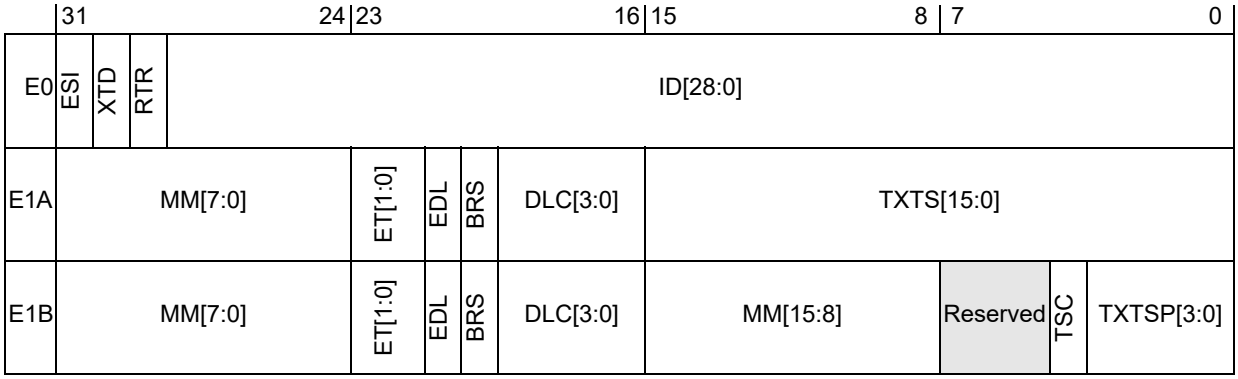


Figure 803. Tx event FIFO element

Table 547. Tx event FIFO element descriptions

Field	Description
E0 bit 31 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 bit 30 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 bit 29 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Table 547. Tx event FIFO element descriptions (continued)

Field	Description
E1A/B bits 31:24 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.
E1A/B bit 23:22 ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1A/B bit 21 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1A/B bit 20 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching
E1A/B bits 19:16 DLC[3:0]	Data Length Code 0-8 CAN + CAN FD: frame with 0-8 data bytes transmitted 9-15 CAN: frame with 8 data bytes transmitted 9-15 CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
E1A bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
E1B bits 15:8 MM[15:8]	Message Marker High byte of wide message marker, written by CPU during Tx buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
E1B bit 4 TSC	Timestamp Captured 0 No timestamp captured 1 Timestamp captured and stored in TSU timestamp register referenced by E1B.TXTSP
E1B bits 3:0 TXTSP[3:0]	Tx Timestamp Pointer Number of TSU timestamp register (TS0..15) where the related timestamp is stored.

47.2.4.4 Standard message ID filter element

Up to 128 filter elements can be configured for 11-bit IDs. When accessing a standard message ID filter element, its address is the Filter List Standard Start Address SIDFC[FLSSA] plus the index of the filter element (0 to 127).

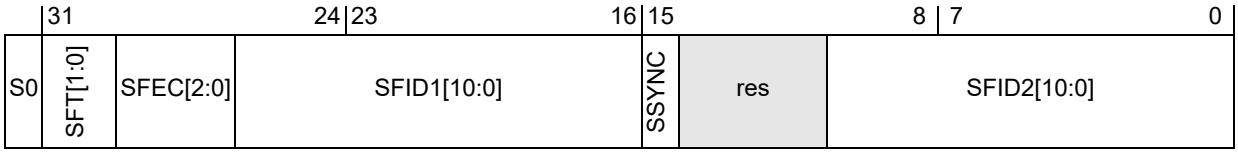


Figure 804. Standard message ID filter element

Table 548. Standard message ID filter element descriptions

Field	Description
Bits 31:30 SFT[1:0]	Standard Filter Type 00 Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1) 01 Dual ID filter for SFID1 or SFID2 10 Classic filter: SFID1 = filter, SFID2 = mask 11 Reserved
Bit 29:27 SFEC[2:0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of 11-bit ID frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = 0x4, 0x5 or 0x6 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches, not intended to be used with Sync messages 100 Set priority if filter matches, not intended to be used with Sync messages, no storage 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx buffer or as debug message, configuration of SFT[1:0] ignored
Bits 26:16 SFID1[10:0]	Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx buffers, Sync messages, or for debug messages this field defines the ID of the message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 15 SSYNC	Standard Sync Message Only evaluated when CCCR.UTSU = 1. When this bit is set and a matching message is received, a pulse with the duration of one MCAN HCLK period is generated at output MCAN TS Rx to signal the reception of a Sync message to the Timestamping Unit (TSU) connected to the M_CAN. 0 Timestamping for the matching Sync message disabled 1 Timestamping for the matching Sync message enabled **Note:** The generation of a pulse at output MCAN TS Rx is independent of whether the matching received message is stored or not. For example, if a message is received for one of the two Rx FIFOs and the FIFO is full and operated in blocking mode, the pulse is generated but the message itself is not stored. In this case the timestamp captured by the TSU does not relate to a message in the M_CAN's Message RAM.

Table 548. Standard message ID filter element descriptions (continued)

Field	Description
Bits 10:0 SFID2[10:0]	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: 1) SFEC = 0x1 t0 0x6 second ID of standard ID filter element 2) SFEC = 0x7 Filter for Rx buffers or for debug messages
	SFID2[10:9] decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx buffer 01 Debug message A 10 Debug message B 11 Debug message C
	SFID2[8:6] is used to control the filter event pins at the extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_CAN host clock period in case the filter matches.
	SFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

47.2.4.5 Extended message ID filter element

Up to 64 filter elements can be configured for 29-bit IDs. When accessing an extended message ID filter element, its address is the Filter List Extended Start Address XIDFC[FLESA] plus two times the index of the filter element (0 to 63).

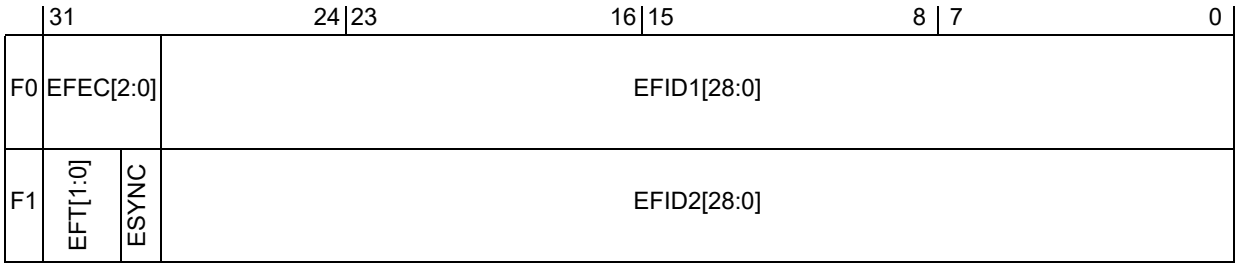


Figure 805. Extended Message ID Filter element

Table 549. Extended Message ID Filter element description

Field	Description
F0 bits 31:29 EFEC[2:0]	Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of 29-bit ID frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches, not intended to be used with Sync messages 100 Set priority if filter matches, not intended to be used with Sync messages, no storage 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx buffer or as debug message, configuration of EFT[1:0] ignored
F0 bits 28:0 EFID1[28:0]	Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx buffers, Sync messages, or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (refer to *Section 47.2.8.1.5: Extended message ID filtering*) is used.
F1 bits 31:30 EFT[1:0]	Extended Filter Type 00 Range filter from EFID1 to EFID1 (EFID2 ≥ EFID1) 01 Dual ID filter for EFID1 or EFID2 10 Classic filter: EFID1 = filter, EFID2 = mask 11 Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), XIDAM mask not applied

Table 549. Extended Message ID Filter element description (continued)

Field	Description
F1 bits 29 ESYNC	Extended Sync Message Only evaluated when CCCR.UTSU = 1. When this bit is set and a matching message is received, a pulse with the duration of one MCAN HCLK period is generated at output MCAN TS Rx to signal the reception of a Sync message to the Timestamping Unit (TSU) connected to the M_CAN. 0 Timestamping for the matching Sync message disabled 1 Timestamping for the matching Sync message enabled **Note:** The generation of a pulse at output MCAN TS Rx is independent of whether the matching received message is stored or not. For example, if a message is received for one of the two Rx FIFOs and the FIFO is full and operated in blocking mode, the pulse is generated but the message itself is not stored. In this case the timestamp captured by the TSU does not relate to a message in the M_CAN's Message RAM.
F1 bits 28:0 EFID2[28:0]	This bit field has a different meaning depending on the configuration of EFEC: 1) EFEC = 0x1 to 0x6 second ID of extended ID filter element 2) EFEC = 0x7 Filter for Rx buffers or for debug messages EFID2[10:9] decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C EFID2[8:6] is used to control the filter event pins at the extension interface. A '1' at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one host clock period in case the filter matches. EFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

47.2.5 M_CAN functional description

47.2.5.1 Operating modes

47.2.5.1.1 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus transmit output is recessive (HIGH). The counters of the Error Management Logic (EML) are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write).

CCCR[CCE] can only be set/reset while CCCR[INIT] = 1. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set

- HPMS–High Priority Message Status
- RXF0S–Rx FIFO 0 Status
- RXF1S–Rx FIFO 1 Status
- TXFQS–Tx FIFO/Queue Status
- TXBRP–Tx Buffer Request Pending
- TXBTO–Tx Buffer Transmission Occurred
- TXBCF–Tx Buffer Cancellation Finished
- TXEFS–Tx Event FIFO Status

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx handler and Rx handler are held in idle state while CCCR[CCE] = ‘1’.

The following registers are only writable while CCCR[CCE] = 0

- TXBAR–Tx buffer add request
- TXBCR–Tx buffer cancellation request

CCCR[TEST] and CCCR.MON can only be set by the host while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’. Both bits may be reset at any time. CCCR.DAR can only be set/reset while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’.

The following sequence can be used as an example code for initializing the CAN module (M_CAN0 in example):

```
-- Setting the INIT bit high to start the initialization
     while (read_data != 0x003){
        write(M_CAN0,0xF,driver.regs.can0_cccr.addr,0x003);
           read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
     };

-- Configuring the time stamp and timeout registers
     write(M_CAN0,0xF,driver.regs.can0_tscc.addr,0x000);
     write(M_CAN0,0xF,driver.regs.can0_tocc.addr,0x000);

-- Configuring the interrupt registers
     write(M_CAN0,0xF,driver.regs.can0_ie.addr,0xFFFFFFFF);
     write(M_CAN0,0xF,driver.regs.can0_ils.addr,0x00000002);
     write(M_CAN0,0xF,driver.regs.can0_ile.addr,0x00000003);

     write(M_CAN0,0xF,driver.regs.can0_gfc.addr,0x000);
     write(M_CAN0,0xF,driver.regs.can0_xidam.addr,0x00FFFFFF);
```

```
-- Writing the start addresses of all the M_CAN Buffers
     generate and drive M_CAN0 can0_rxf0c keeping {
                                              .f0sa == 0x100;
                                              .f0s  == 0x03;
                                              .f0wm == 0x01;
                                         };
     generate and drive M_CAN0 can0_rxf1c keeping {
                                                 .f1sa == 0x200;
                                                 .f1s  == 0x03;
                                                 .f1wm == 0x01;
                                                 };
     generate and drive M_CAN0 can0_sidfc keeping {
                                                 .flssa == 0x000;
                                                 .lss   == 0x80;
                                                 };
     generate and drive M_CAN0 can0_xidfc keeping {
                                                 .flesa == 0x080;
                                                 .lse   == 0x40;
                                                 };
     generate and drive M_CAN0 can0_txbc  keeping {
                                                 .tbsa == 0x300;
                                                 .ndtb == 0x020;
                                                 .tfqs == 0x1D;
                                                 };
     generate and drive M_CAN0 can0_txefc keeping {
                                              .efsa == 0x380;
                                              .efs == 0x020;
                                              .efwm == 0x00;
                                              };
-- Writing to the RAM for filter configuration
=================================================
-- Writing to TX Buffer
     write(RAM,0xF,0xC00,0x04040000);
     write(RAM,0xF,0xC04,0x10830000);
     write(RAM,0xF,0xC08,0x00010001);
     write(RAM,0xF,0xC10,0x50000002);
     write(RAM,0xF,0xC14,0x11830000);
     write(RAM,0xF,0xC18,0x00070102);

-- Writing to s-filter
     write(RAM,0xF,0x000,0x58020004);
     write(RAM,0xF,0x004,0x08000005);
     write(RAM,0xF,0x008,0x58060007);
     write(RAM,0xF,0x00C,0x08030008);
     write(RAM,0xF,0x010,0x880E07FB);
```

```
      write(RAM,0xF,0x014,0x980D07FE);
      write(RAM,0xF,0x018,0x68100100);
      write(RAM,0xF,0x01C,0x70120200);
      write(RAM,0xF,0x020,0x981F07F6);
      write(RAM,0xF,0x024,0x50140014);
      write(RAM,0xF,0x028,0x58160017);
      write(RAM,0xF,0x02C,0x08110019);
      write(RAM,0xF,0x030,0x4A22001A);
      write(RAM,0xF,0x034,0xA01A07FF);
      write(RAM,0xF,0x038,0x5BFF07E3);
      write(RAM,0xF,0x03C,0x080007FF);

-- Writing to x-filter
      write(RAM,0xF,0x200,0x201ABCD9);
      write(RAM,0xF,0x204,0x001ABCE5);
      write(RAM,0xF,0x208,0x403BCDEF);
      write(RAM,0xF,0x20C,0x003BCDEF);
      write(RAM,0xF,0x210,0x605CDEF2);
      write(RAM,0xF,0x214,0x405CDEF0);
      write(RAM,0xF,0x218,0x407DEEFF);
      write(RAM,0xF,0x21C,0x007DEF01);
      write(RAM,0xF,0x220,0x209ABCDE);
      write(RAM,0xF,0x224,0x009ABCDF);
      write(RAM,0xF,0x228,0xA0BBCDEF);
      write(RAM,0xF,0x22C,0x40BBCDEE);
      write(RAM,0xF,0x230,0x20DCDEF0);
      write(RAM,0xF,0x234,0x9FFFFFFF);
      write(RAM,0xF,0x238,0x40000000);
      write(RAM,0xF,0x23C,0x1FFFFFFF);
================================================
write(M_CAN0,0xF,driver.regs.can0_nbtp.addr,0x00002303);

      print "Configuration done";
      read_data = 0x01;
      while (read_data != 0x00){
         write(M_CAN0,0xF,driver.regs.can0_cccr.addr, 0x000);
            wait [1];
            read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
         };

      print "Initialization done";
```

47.2.5.2 Normal operation

Once the M_CAN is initialized and CCCR[INIT] is reset to zero, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, the received messages including message ID and DLC are stored into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx buffers or a Tx FIFO, or both, or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

47.2.5.3 CAN FD operation

There are two variants in the CAN FD frame transmission, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers is now decoded as FDF bit. FDF = *recessive* signifies a CAN FD frame, FDF = *dominant* signifies a classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = *dominant* and BRS = *recessive*. The coding of res = *recessive* is reserved for future expansion of the protocol. In case the M_CAN receives a frame with FDF = *recessive* and res = *recessive*, it signals a protocol exception event by setting bit PSR[PXE]. When Protocol Exception Handling is enabled (CCCR[PXHD] = 0), this causes the operation state to change from receiver (PSR[ACT] = 0x2) to Integrating (PSR[ACT] = 0x0) at the next sample point. In case protocol exception handling is disabled (CCCR[PXHD] = 1), the M_CAN treats a *recessive* res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming CCCR[FDOE]. In case CCCR.FDOE = 1, transmission and reception of CAN FD frames are enabled. Transmission and reception of classic CAN frames are always possible. The transmission of a CAN FD frame or of a classic CAN frame can be configured via bit FDF in the respective Tx buffer element. With CCCR[FDOE] = 0, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx buffer element is set. CCCR[FDOE] and CCCR[BRSE] can only be changed while CCCR[INIT] and CCCR[CCE] are both set.

With CCCR[FDOE] = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in classic CAN format. With CCCR[FDOE] = 1 and CCCR[BRSE] = 0, only bit FDF of a Tx buffer element is evaluated. With CCCR[FDOE] = 1 and CCCR[BRSE] = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significantly higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions
- During system startup all nodes are transmitting classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Then all nodes switch back to classic CAN communication

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to *Table 550*.

Table 550. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing is switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is *recessive*. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the Nominal Bit Timing and Prescaler register NBTP. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing and Prescaler register DBTP. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (CAN clock). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted *recessive*, else it is transmitted *dominant*.

47.2.5.4 Transceiver delay compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via CAN Tx output pin the M_CAN the transmitted data from its local CAN transceiver via CAN Rx input pin. The received data is delayed by the transmitter delay. In case this delay is greater than NTSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During arbitration the transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit DBTP.TDC.phase the delay compensation is always disabled.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the M_CAN's Tx output through the transceiver to the CAN Rx input plus the transmitter delay compensation offset as configured by TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.

PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

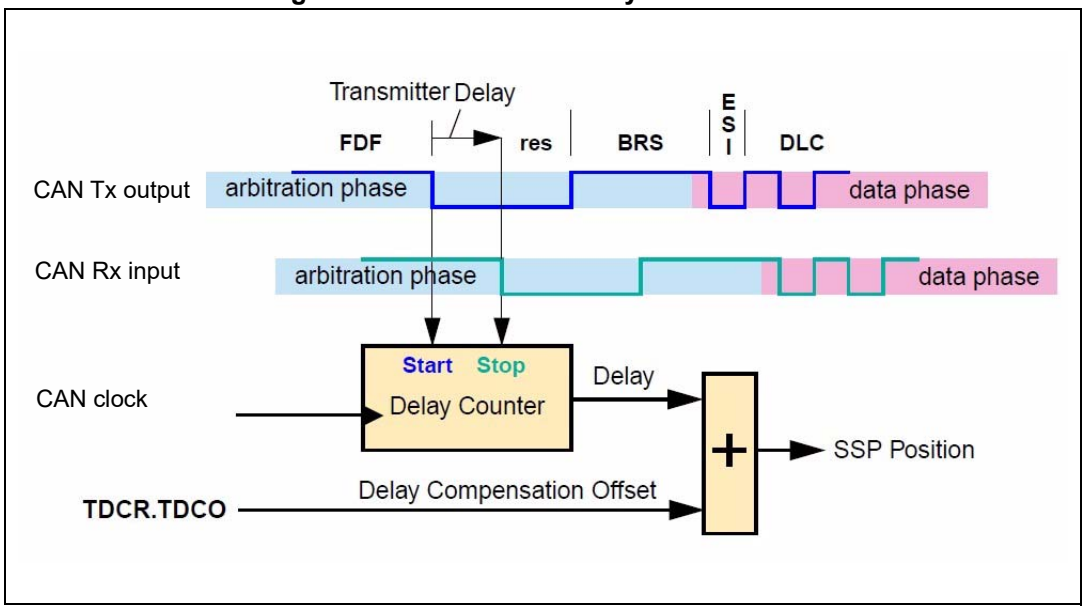
The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_CAN:

- The sum of the measured delay from CAN Tx output to CAN Rx input and the configured transmitter delay compensation offset TDCR.TDCO has to be less than 6 bits times in the data phase.
- The sum of the measured delay from CAN Tx output to CAN Rx input and the configured transmitter delay compensation offset TDCR.TDCO has to be less than or equal to 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs.

Figure 806 describes the measurement of transceiver delay compensation.

If transmitter delay compensation is enabled by programming DBTP[TDC] = 1, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input CAN Rx input of the transmitter. The resolution of this measurement is 1 mtq.

Figure 806. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a too early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on CAN Rx input, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCR.TDCF AND CAN Rx input is low.

Note: The M_CAN always performs transmitter delay measurement, it does not support the configuration of a static (fixed) transmitter delay.

47.2.5.5 Restricted operation mode

In restricted operation mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. The Host can set the M_CAN into Restricted Operation mode by setting bit CCCR.ASM. The bit can only be set by the Host when both CCCR.CCE and CCCR.INIT are set to 1. The bit can be reset by the host at any time.

Restricted operation mode is automatically entered when the Tx handler was not able to read data from the message RAM in time. To leave Restricted operation mode, the host CPU has to reset CCCR.ASM.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

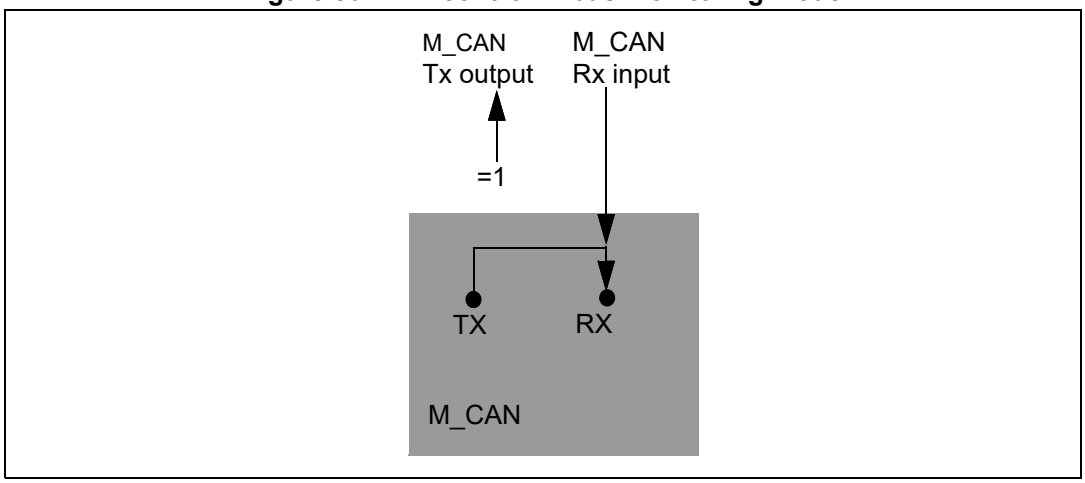
47.2.5.6 Bus monitoring mode

The M_CAN is set in bus monitoring mode by programming CCCR[MON] to 1. In Bus Monitoring Mode (refer to ISO11898-1, 10.12 Bus monitoring), the M_CAN is able to receive

valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In bus monitoring mode, register TXBRP is held in reset state. The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits.

Figure 807 shows the connection of M_CAN Tx and Rx signals to the M_CAN in Bus monitoring mode.

Figure 807. Pin control in bus monitoring mode



47.2.5.7 Disabled automatic retransmission

According to the CAN specification (refer to ISO11898-1, 6.3.3 recovery management), the M_CAN provides the means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

47.2.5.7.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they started on the CAN bus. A Tx buffer's Tx Request Pending bit TXBRP[TRPx] is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with Event Type ET = 0x2 (transmission in spite of cancellation).

47.2.5.8 Power down (sleep mode)

The M_CAN can be set into power down mode controlled by input signal clock stop request or via CC Control Register CCCR[CSR]. As long as the clock stop request signal is active, bit CCCR[CSR] is read as 1.

When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] has no effect. Now the module clock inputs (CAN clock and host clock) may be switched off. To leave power down mode, the application has to turn on the module clocks before resetting signal clock stop request signal resp. CC Control Register flag CCCR[CSR]. The M_CAN acknowledges this by resetting clock stop acknowledge output signal and resetting CCCR[CSA] to 1. Before the module clocks CAN host clock and CAN clock are switched off, further register accesses can be made except for CCCR[INIT] which is held at 1.

Note: In case of a heavily disturbed CAN bus, it may happen that idle state is never reached and CCCR[INIT] is therefore not set by the M_CAN. This situation can be detected by polling PSR[ACT]. In case the M_CAN does not enter idle state, the software can write CCCR[INIT] = 1 which stops CAN communication of the M_CAN immediately, regardless of whether there is a transmission/reception ongoing or not.

To leave power down mode, the application has to turn on the module clocks before resetting signal clock stop request signal resp. CC Control Register flag CCCR[CSR]. The M_CAN acknowledges this by resetting clock stop acknowledge output signal and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

47.2.5.9 Test modes

To enable write access to register TEST (refer to *Section 47.2.3.4: Test register (TEST)*), bit CCCR[TEST] has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN Tx output pin by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN sample point signal to monitor the M_CAN's bit timing and it can drive constant dominant or recessive values. The actual value at M_CAN Rx pin can be read from TEST[RX]. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN clock and host clock domain, there may be a delay of several host clock periods between writing to TEST[TX] until the new configuration is visible at output Tx pin. This applies also when reading input Rx pin via TEST[RX].

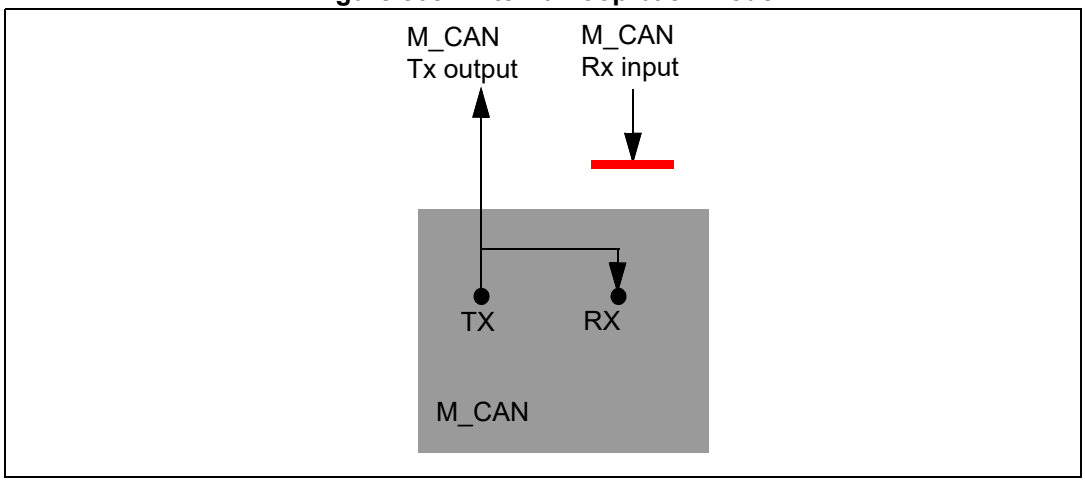
Note: Test modes must be used for production tests or self test only. The software control for Tx pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

47.2.5.9.1 External loop back mode

The M_CAN can be set in external loop back mode by programming TEST.LBCK to 1. In Loop Back mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 808* shows the connection of Tx and Rx signals to the M_CAN in external loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop back mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN Rx input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the CAN Tx output pin.

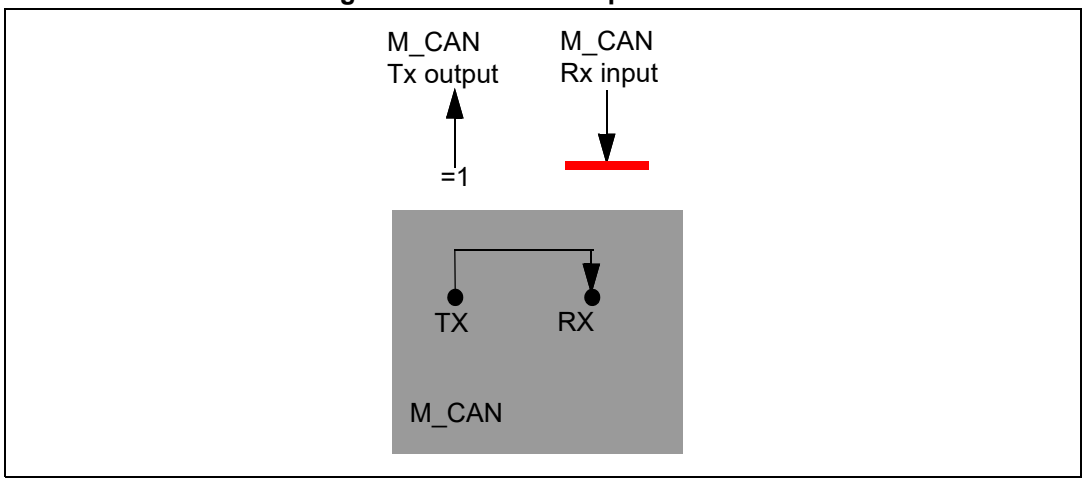
Figure 808. External loop back mode



47.2.5.9.2 Internal loop back mode

Internal loop back mode is entered by programming bits TEST[LBCK] and CCCR[MON] to 1. This mode can be used for a “Hot Selftest”, meaning the M_CAN can be tested without affecting a running CAN system connected to the CAN Tx output and Rx input pins. In this mode M_CAN Tx pin is disconnected from the M_CAN and M_CAN Tx pin is held recessive. *Figure 809* shows the connection of M_CAN Tx pin and Rx to the M_CAN in case of internal loop back mode.

Figure 809. Internal loop back mode



47.2.6 Timestamp generation

47.2.6.1 Internal timestamp generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler TSCC[TCP] can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via TSCV[TSC]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On Start Of Frame (SOF) reception/transmission the counter value is captured and stored into the timestamp section of an Rx buffer / Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming bit TSCC[TSS], the external 16-bit timebase vector input of the M_CAN can be captured as timestamp instead of the internal 16-bit counter.

47.2.6.2 Timestamp generation by use of an external TSU

An external Time Stamping Unit (TSU) for generation of 32-bit timestamps according to CiA 603 can be connected to the M_CAN's TSU interface. External timestamping is enabled when CCCR.[UTSU] = 1.

To filter for Sync messages a standard or extended filter element has to be configured with the Sync message's ID and bit SSYNC resp. ESYNC set to 1.

At the End Of Frame (EOF) reception/transmission, the timestamp is captured by the TSU, controlled by the M_CAN's timestamp Rx and Tx output signals. The number of the TSU's timestamp register which holds the captured timestamp is signalled to the M_CAN by MCAN TSP[2:0] and is stored into the related Rx buffer / Rx FIFO element (R1B.RXTSP[2:0]) resp. Tx event FIFO element (E1B.TXTSP[2:0]). For a detailed description refer to TSU chapter.

47.2.7 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the M_CAN supplies a 16-bit timeout counter. It operates as down-counter and uses the same prescaler controlled by TSCC.TCP as the timestamp counter. The timeout counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The timeout

counter can only be started while CCCR[INIT] = 0. It is stopped when CCCR[INIT] = 1, for example when the M_CAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In continuous mode, the counter is immediately restarted at TOCC[TOP].

Note: The clock signal for the timeout counter is derived from the CAN core's sample point signal. Therefore the point in time where the timeout counter is decremented may vary due to the synchronization/re-synchronization mechanism of the CAN core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

47.2.8 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get indexes.

47.2.8.1 Acceptance filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as:
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled/disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the message handler starts writing the received message data in portions of 32-bit to the matching Rx buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (such as CRC error), this message is discarded with the following impact on the affected Rx buffer or Rx FIFO:

Rx buffer

New Data flag of matching Rx buffer is not set, but Rx buffer (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.DLEC.

Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Section 47.2.8.2.2: Rx FIFO overwrite mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into a Rx Buffer, the unmodified received identifier is stored independently of the filter(s) used. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

47.2.8.1.1 Range filter

The filter matches for all received frames with Message IDs in the range defined by SFID1/SFID2 resp. EFID1/EFID2. There are two possibilities when range filtering is used together with extended frames:

- EFT = 0x0: The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied.
- EFT = 0x3: The Extended ID AND Mask (XIDAM) is not used for range filtering.

47.2.8.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SFID1 = SFID2 resp. EFID1 =EFID2.

47.2.8.1.3 Classic bit mask filter

Classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SFID1/EFID2 is used as message ID filter, while SFID2/EFID2 is used as filter mask.

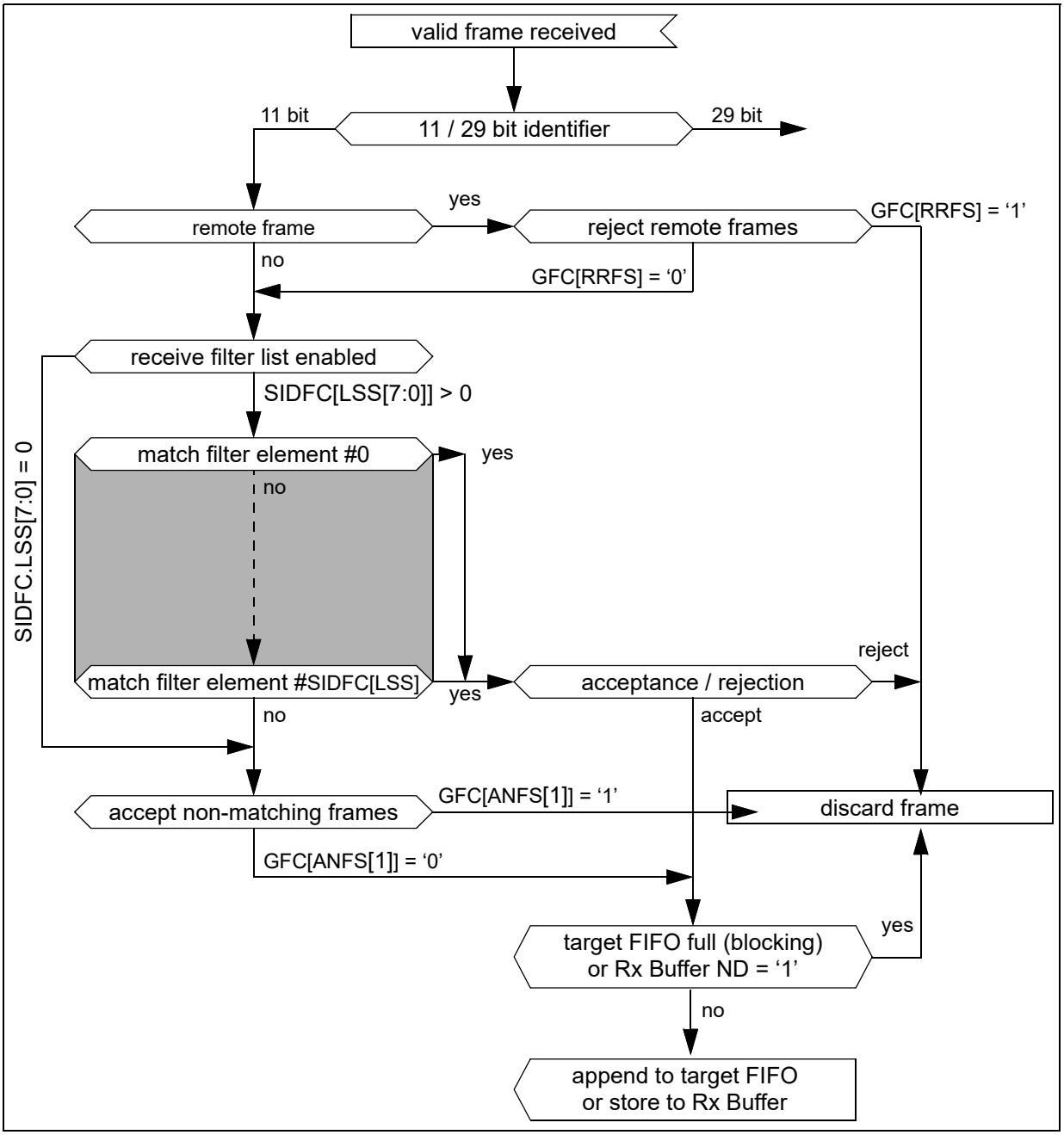
A zero bit at the filter mask masks out the corresponding bit position of the configured ID filter, for example the value of the received message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received message ID where the corresponding mask bits are one are relevant for acceptance filtering. In case all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

47.2.8.1.4 Standard message ID filtering

Figure 804 shows the flow for standard message ID (11-bit Identifier) filtering. The standard message ID filter element is described in *Section 47.2.4.4*.

Controlled by the Global Filter Configuration (GFC) and the Standard ID Filter Configuration (SIDFC) message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 810. Standard message ID filter path



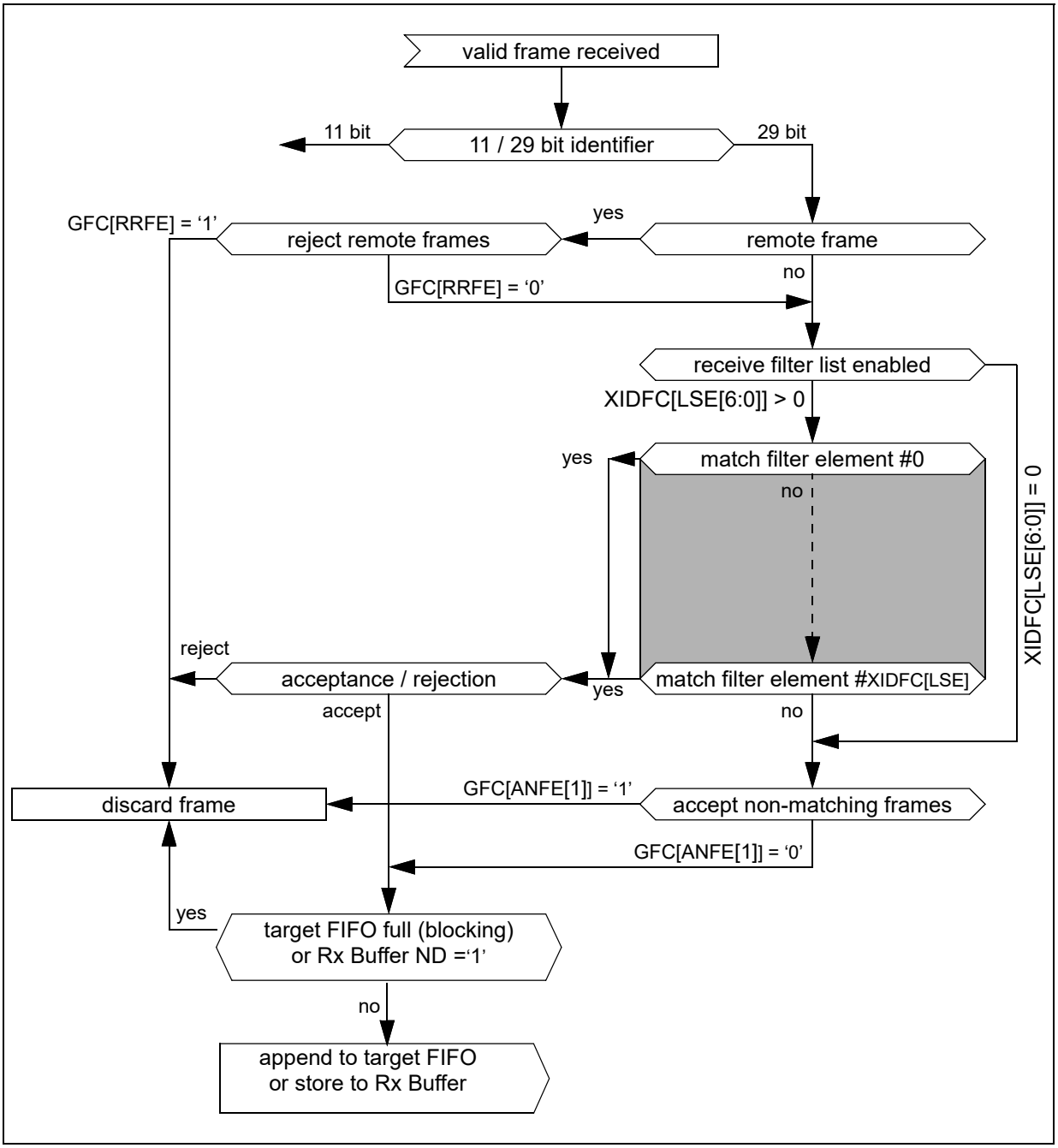
47.2.8.1.5 Extended message ID filtering

Figure 811 below shows the flow for extended message ID (29-bit Identifier) filtering. The extended message ID filter element is described in *Section 47.2.3.21: Extended ID filter configuration register (XIDFC)*.

Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements which is controlled by the Global Filter Configuration (GFC) and the Extended ID Filter Configuration (XIDFC)

message ID. The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

Figure 811. Extended message ID filter path



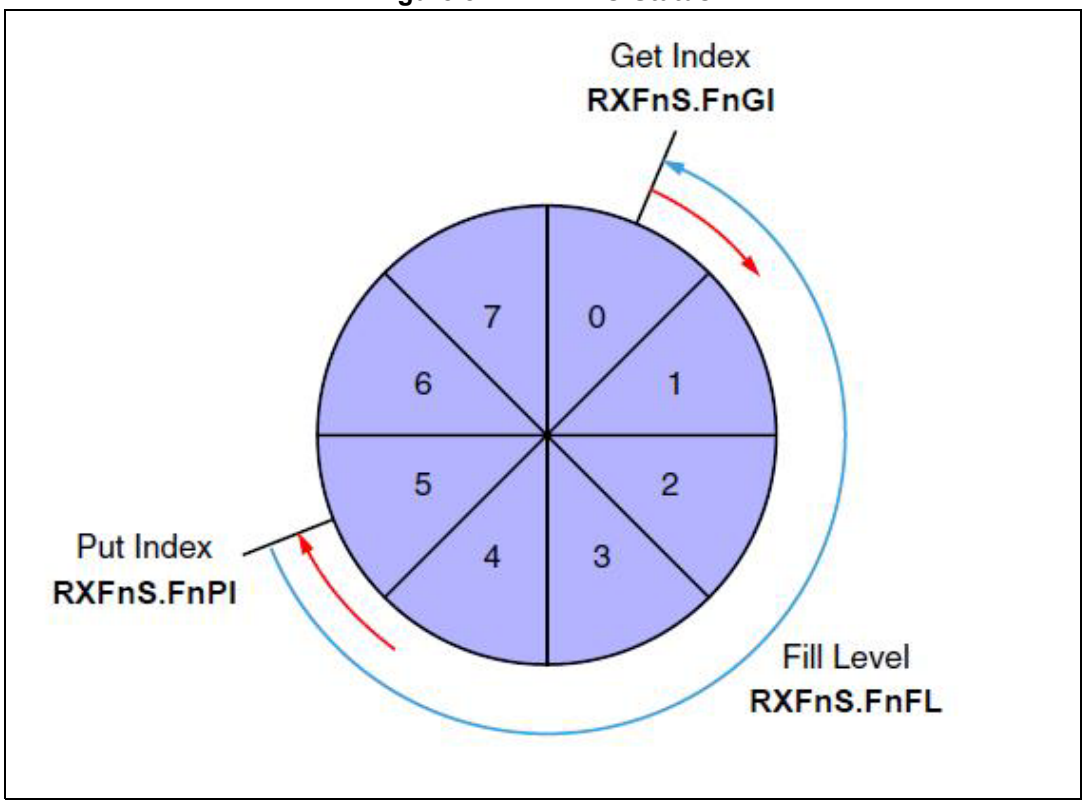
47.2.8.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 refer to *Section 47.2.8.1: Acceptance filtering*. The Rx FIFO element is described in *Section 47.2.4.1: Rx buffer and FIFO element*.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag IR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signaled by RXFnS.FnF. In addition interrupt flag IR.RFnF is set.

Figure 812. Rx FIFO Status



When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI × FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 551. Rx buffer / FIFO element size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10

Table 551. Rx buffer / FIFO element size (continued)

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
110	48	14
111	64	18

47.2.8.2.1 Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = 0. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signaled by RXFnS.FnF = 1. In addition interrupt flag IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = 1. In addition interrupt flag IR.RFnL is set.

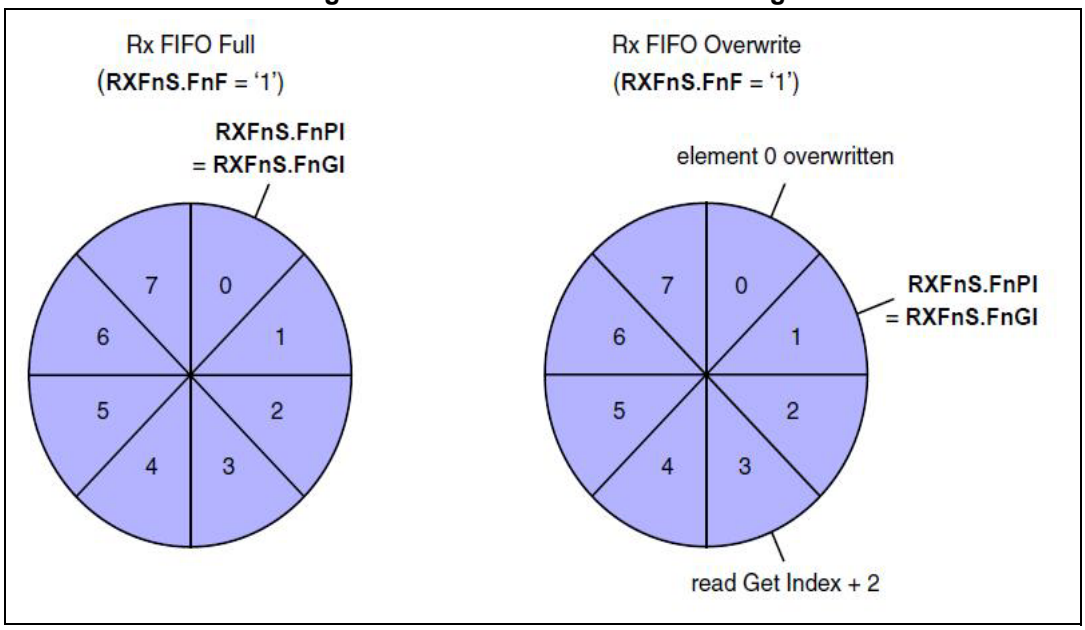
47.2.8.2.2 Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = 1.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signaled by RXFnS.FnF = 1, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get indexes are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements must start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. *Figure 813* shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

Figure 813. Rx FIFO overflow handling



After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = 0).

47.2.9 Dedicated Rx buffers

The M_CAN supports up to 64 dedicated Rx buffers. The start address of the dedicated Rx buffer section is configured via RXBC.RBSA.

For each Rx buffer a standard or extended message ID filter element with SFEC/EFEC = 0x7 and SFID2 / EFID2[10:9] = 0x0 has to be configured (refer to *Section 47.2.4.4: Standard message ID filter element* and *Section 47.2.4.5: Extended message ID filter element*).

After a received message has been accepted by a filter element, the message is stored into the Rx buffer in the message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (message stored in dedicated Rx buffer) in the interrupt register is set.

Table 552. Example filter configuration for Rx buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message 1	00	00 0000
1	ID debug message 2	00	00 0001
2	ID debug message 3	00	00 0010

After the last word of a matching received message has been written to the message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the new data flag is set,

the respective Rx Buffer is locked against updates from received matching frames. The new data flags have to be reset by the host by writing a 1 to the respective bit position.

While an Rx buffer's new data flag is set, a message ID filter element referencing this specific Rx buffer does not match, causing the acceptance filtering to continue. Following message ID filter elements may cause the received message to be stored into another Rx buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

47.2.9.1 Rx buffer handling

- Reset interrupt flag IR.DRX
- Read new data registers
- Read messages from message RAM
- Reset new data flags of processed messages

47.2.10 Debug on CAN support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (for example #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (refer to *Section 47.2.4.1: Rx buffer and FIFO element*).

Advantage: Fixed start address for the DMA transfers (relative to RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC/EFEC = 0x7 have to be set up. Messages matching these filter elements are stored into the Rx buffers addressed by SFID2/EFID2[5:0].

After message C has been stored, the DMA request output is activated and the three messages can be read from the message RAM under DMA control. The RAM words holding the debug messages is not changed by the M_CAN while DMA request output is activated. The behavior is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets the DMA acknowledge input. This resets the DMA request output. Now the M_CAN is prepared to receive the next set of debug messages.

47.2.10.1 Filtering for debug messages

Filtering for debug messages is done by configuring one standard/extended message ID filter element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC/EFEC has to be programmed to 0x7. In this case fields SFID1/SFID2 and EFID1/EFID2 have a different meaning. While SFID2/EFID2[10:9] controls the debug message handling state machine, SFID2/EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective new data flag nor IR.DRX are set. The reception of debug messages can be monitored via RXF1S.DMS.

Table 553. Example filter configuration for debug messages

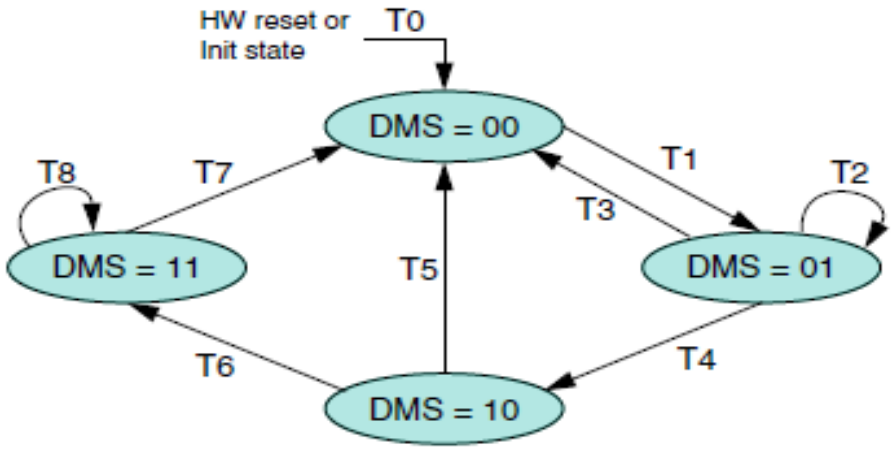
Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

47.2.10.2 Debug message handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signaled via RXF1S.DMS.

Figure 814. Debug message handling state machine



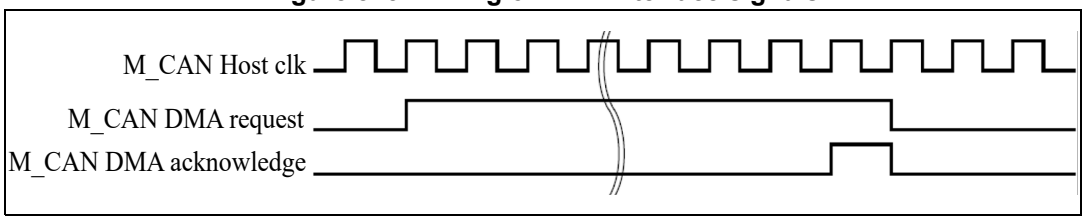
T0: reset m_cam_dma_req output, enable reception of debug messages A, B, and C
T1: reception of debug message A
T2: reception of debug message A
T3: reception of debug message C
T4: reception of debug message B
T5: reception of debug messages A, B
T6: reception of debug message C
T7: DMA transfer completed
T8: reception of debug message A,B,C (message rejected)

47.2.10.3 Interface to DMA controller

When all three debug messages A, B, C have been received in the correct order, M_CAN DMA request signal is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C is not changed by the M_CAN while M_CAN DMA request signal is active.

After the transfer of the received messages has completed the DMA unit activates the M_CAN DMA ACK signal. This resets M_CAN DMA request signal. The debug message handling state machine enters idle state (DMS = 0x0) and waits for reception of the next debug messages.

Figure 815. Timing of DMA interface signals



Note: If the DMA unit activates input M_CAN DMA acknowledge signal before the DMA transfer has completed, the Rx buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.

47.2.11 Tx handling

The Tx handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the Put and Get indexes, and the Tx event FIFO. Up to 32 Tx buffers can be set up for message transmission. The Tx buffer element is described in the *Note: on page 1779*.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

The Tx handler starts a Tx scan of the message RAM to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx Buffer Request Pending register TXBRP is updated. TXBRP is updated when a transmission finishes, or after the host has requested a transmission by writing to TXBAR, or when the host has canceled a pending transmission by writing to TXBCR.

When there is a new scan-result, the temporary buffers holding the preloaded first four RAM words of the two messages with the highest Tx priority are updated. The time required for a Tx scan and the preloading depends on the host clock frequency, on the number of configured Tx buffers, and on the number of M_CANs connected the message RAM.

As the Tx scan of the message RAM takes some time, the following scenarios are possible:

- Temporary buffer preloaded before ongoing transmission/reception finished: Transmission of preloaded Tx message starts at first opportunity
- Preloading of temporary buffer not completed before ongoing transmission/reception finished: Tx message cannot be started at first opportunity. In this case another node may start a transmission without having to arbitrate against the transmit message of the M_CAN. If this other message has lower priority, that may be regarded as an outer priority inversion.

47.2.11.1 Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined

messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If for example CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, after the first successful message transmission, it waits for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit CCCR.TXP. If the bit is set, the M_CAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (CCCR.TXP = 0).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

47.2.11.2 Dedicated Tx buffers

Dedicated Tx buffers are intended for message transmission under complete control of the host CPU. Each dedicated Tx buffer is configured with a specific message ID. These Tx buffers must be requested in ascending order with the lowest buffer number first. Alternatively, all Tx buffers configured with the same Message ID can be requested simultaneously by a single write access to TXBAR.

If the data section has been updated, a transmission is requested by an Add request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their message ID.

A Dedicated Tx buffer allocates element size 32-bit words in the message RAM (refer to *Table 554*). Therefore the start address of a dedicated Tx buffer in the message RAM is calculated by adding transmit buffer index (0 to 31) × element size to the Tx Buffer Start Address TXBC.TBSA.

Table 554. Tx buffer / FIFO / queue element size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10

Table 554. Tx buffer / FIFO / queue element size (continued)

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
110	48	14
111	64	18

47.2.11.3 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get index TXFQS[TFGI]. After each transmission the Get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put index reaches the Get index, Tx FIFO Full (TXFQS[TFQF] = 1) is signaled. In this case no further messages must be written to the Tx FIFO until the next message has been transmitted and the Get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a 1 to the TXBAR bit related to the Tx buffer referenced by the Tx FIFOs Put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the Put index. The transmissions are then requested via TXBAR. The Put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level. When a transmission request for the Tx Buffer referenced by the Get index is canceled, the Get index is incremented to the next Tx buffer with pending transmission request and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the Get index and the FIFO free level remain unchanged.

A Tx FIFO element allocates element size 32-bit words in the message RAM (refer to *Table 554*). Therefore the start address of the next available (free) Tx FIFO buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0 to 31) × element size to the Tx buffer start address TXBC.TBSA.

47.2.11.4 Tx queue

Tx queue operation is configured by programming TXBC[TFQM] to 1. The messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). If multiple Tx Queue buffers are configured with the same Message ID, the transmission order depends on the numbers of the buffers where the messages were stored for transmission. As these buffer numbers depend on the current states of the PUT index, a prediction of the transmission order is not possible. New messages have to be written to the Tx buffer referenced by the Put index TXFQS[TFQPI]. The Put Index always points to that free buffer of the Tx Queue with the lowest buffer number. If the Tx queue is full (TXFQS[TFQF] = 1), the Put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

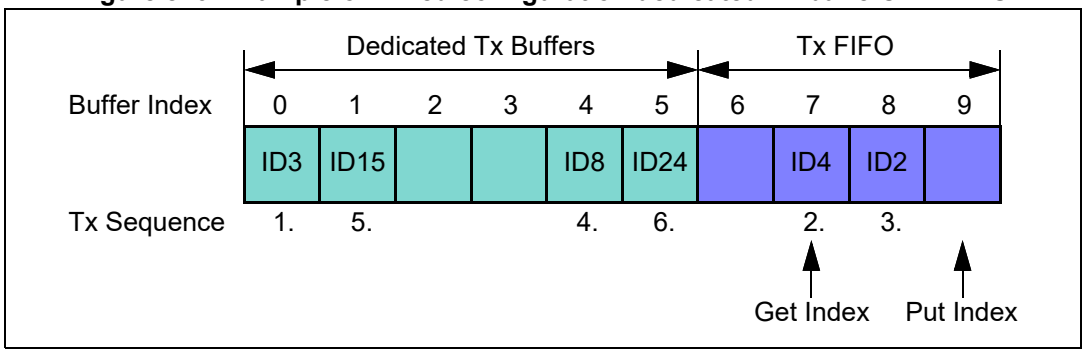
The application can use register TXBRP instead of the Put Index and can place messages to any Tx Buffer without pending transmission request.

A Tx queue buffer allocates element size 32-bit words in the message RAM (refer to *Table 554*). Therefore the start address of the next available (free) Tx queue buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0 to 31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

47.2.11.5 Mixed dedicated Tx buffers /Tx FIFO

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx FIFO. The number of dedicated Tx buffers is configured by TXBC[NDTB]. The number of Tx buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only dedicated Tx buffers are used.

Figure 816. Example of mixed configuration dedicated Tx buffers /Tx FIFO



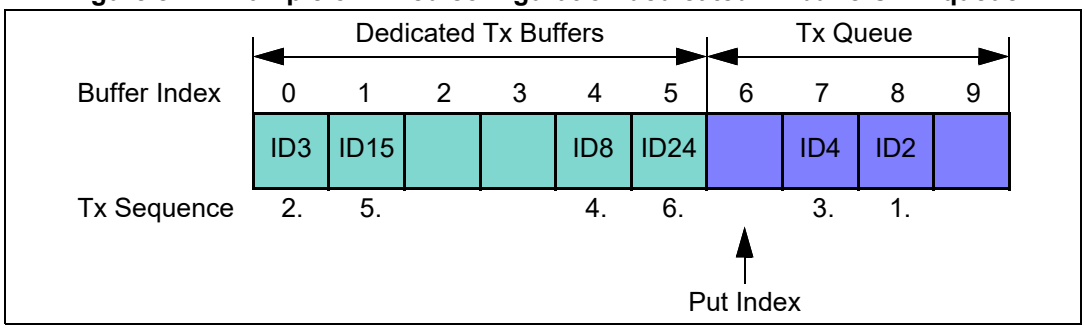
Tx prioritization:

- Scan dedicated Tx buffers and oldest pending Tx FIFO buffer (referenced by TXFS[TFGI])
- Buffer with lowest message ID gets highest priority and is transmitted next

47.2.11.6 Mixed dedicated Tx buffers /Tx queue

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx queue. The number of dedicated Tx buffers is configured by TXBC[NDTB]. The number of Tx queue buffers is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only dedicated Tx buffers are used.

Figure 817. Example of mixed configuration dedicated Tx buffers /Tx queue



Tx prioritization:

- Scan all Tx buffers with activated transmission request
- Tx Buffer with lowest message ID gets highest priority and is transmitted next

47.2.11.7 Transmit cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx buffer or a Tx queue buffer the host has to write a 1 to the corresponding bit position (= number of Tx buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to 1.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

47.2.11.8 Tx event handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

CCCR[WMM] can configure whether an 8-bit message marker or a 16-bit wide message marker is used.

Note: With CCCR.WMM = 1, internal timestamping is disabled (refer to Section 47.2.4.3: Tx event FIFO element).

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx event FIFO element is described in the note "*Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 2 to 17) are used for storage of a CAN message's data field.*". When a Tx Event FIFO full condition is signaled by IR.TEFF, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx Event FIFO Get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx event FIFO overflow, the Tx event FIFO watermark can be used. When the Tx event FIFO fill level reaches the Tx event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx event FIFO Get index TXEFS[EFGI] has to be added to the Tx event FIFO start address TXEFC[EFSA].

47.2.12 FIFO acknowledge handling

The Get Indexes of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index (refer to *Section 47.2.3.28: Rx FIFO 0 acknowledge register (RXF0A)* and *Section 47.2.3.46: Tx Event FIFO Acknowledge Register (TXEFA)*). Writing to the FIFO acknowledge index sets the FIFO Get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the Get index), this Get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value: Index of the last element read), to update the FIFOs Get index.

Due to the fact that the CPU has free access to the M_CAN's message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case the FIFOs acknowledge index must not be written because this would set the Get index to a wrong position and also alters the FIFOs Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The M_CAN does not check for erroneous values.

48 DMA interface unit (DMU)

48.1 Overview

The DMU is an extension-module for the M_CAN which provides an additional interface that allows simple enqueuing and dequeuing to TX and RX FIFOs of the M_CAN by linear write or read accesses, in order to be able to offload these tasks from CPU to standard DMA controllers.

48.1.1 M_CAN message handling (without DMU)

The M_CAN has an associated Message RAM (MRAM) which contains the CAN messages, organized for example in FIFOs. The controlling of that data is done by the M_CAN in conjunction with several registers which provide status and means for handling via CPU. To interchange messages between CRAM and MRAM without the DMU add-on, the CPU has to process the following tasks:

- React on interrupt of M_CAN and check status registers (Alternatively poll M_CAN interrupt / status registers)
- Transfer CAN message between MRAM (Message RAM) and CRAM (micro-controller RAM)
- Acknowledge transfer at M_CAN registers

This complex interaction cannot be handled by standard DMA controllers.

48.1.2 Message handling with DMU

The basic idea of the DMU is to define Virtual Buffers for CAN message transfer. The DMU redirects accesses to these Virtual Buffers dynamically to the MRAM. Redirections are controlled by the FIFO pointers of the M_CAN. The DMU supports message transfers from the CRAM to the TX FIFO/Queue and from the RX FIFOs / TX Event FIFO to the CRAM.

Table 555. Supported M_CAN message buffers

MRAM	Direction	Condition	DMU request signal
TX FIFO/Queue	CRAM →MRAM	Not full	dmu_txr
RX FIFO 0	MRAM →CRAM	Not empty	dmu_rx0r
RX FIFO 1	MRAM →CRAM	Not empty	dmu_rx1r
TX Event FIFO	MRAM →CRAM	Not empty	dmu_txer

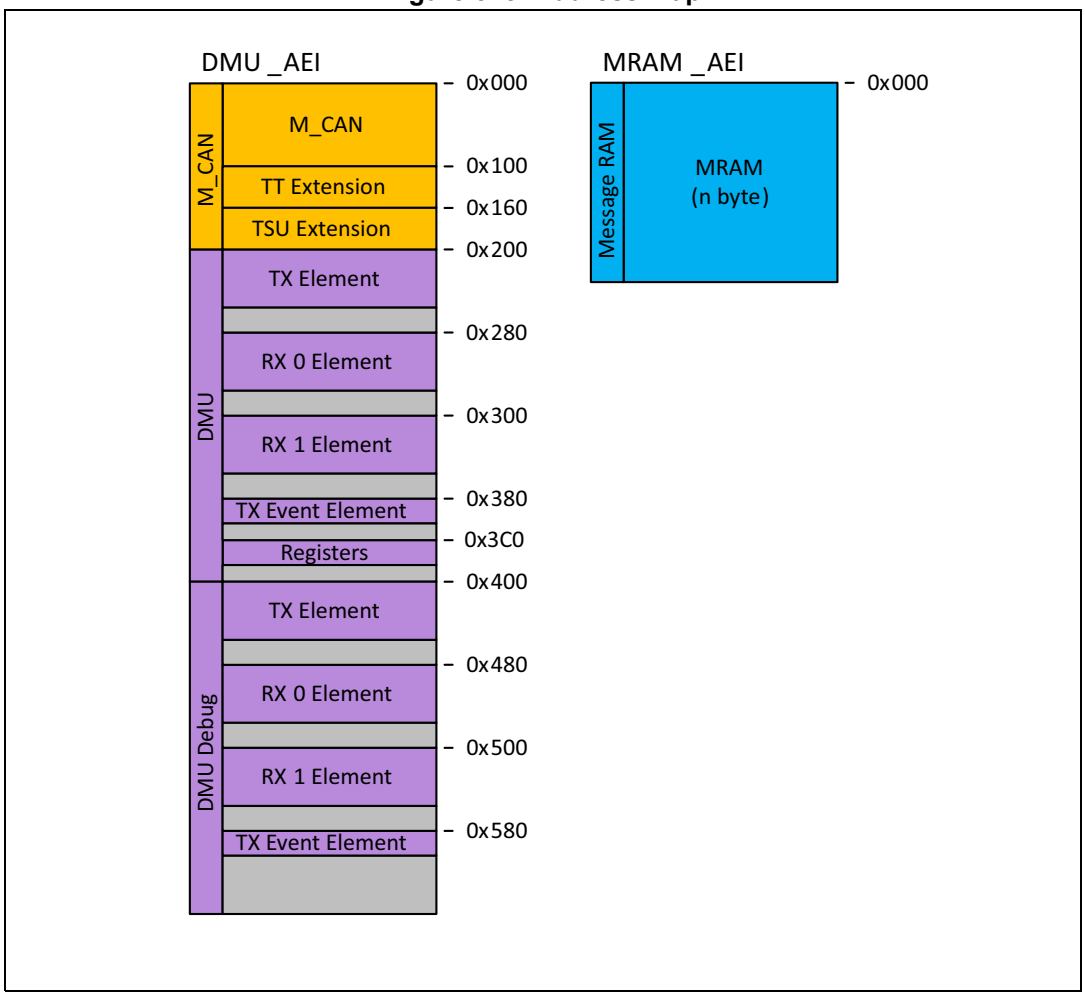
The M_CAN FIFOs have to be configured as described in M_CAN chapter, for example base addresses and data field sizes. When the M_CAN is setup and running, the CAN messages have to be interchanged via dedicated DMU Virtual Buffers as shown in *Figure 818*. These sections are bound to the Tx FIFO respective Rx FIFO head elements, that are the elements, pointed by M_CAN PUT_INDEX respective GET_INDEX. The DMU just redirects the accesses to the MRAM.

The data structure of the FIFO elements is described in *Section 47.2.4: Message RAM* from M_CAN chapter. Each DMU Virtual Buffer is dimensioned to the maximal M_CAN element size, that is 72 byte for Tx and Rx FIFO elements respective 8 byte for Tx Event FIFO element.

Transfer requests to the external DMA controller are indicated by separate DMU Request signals, one for each Virtual Buffer, refer to *Table 555*. On active request, the DMA controller may interchange data, by linear access to the dedicated Virtual Buffer. With this linear transfer, everything internal is implicitly handled, for example that a Tx message is sent by the M_CAN, or a Rx FIFO element is acknowledged and disengaged at the M_CAN.

For software debugging purposes, the Virtual Buffers can be accessed via debug section, without affecting the data flow of normal operation, see *Section 48.3.8: Software debugging support*.

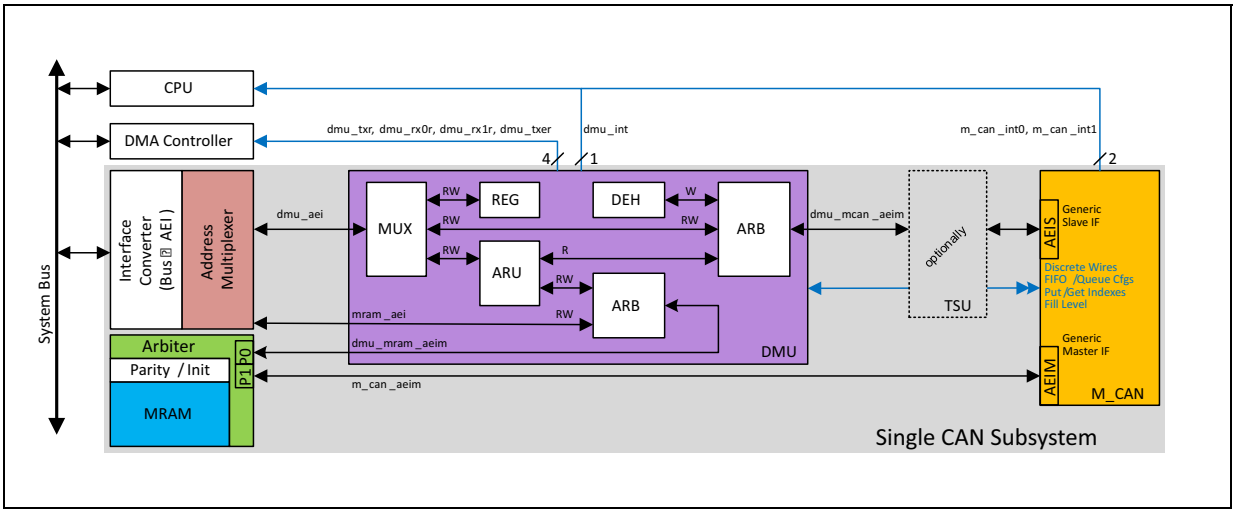
Figure 818. Address map



48.1.3 Block diagram

The following figure shows the internal structure of the DMU and exemplary the integration into a SoC together with M_CAN and optional TSU.

Figure 819. DMU block diagram



48.2 Memory map and register definitions

48.2.1 Hardware reset description

After hardware reset, the registers of the DMU contain the reset values listed in *Table 556*. For the initial M_CAN configuration, the bit M_CAN CCCR.CCE has to be set, and afterwards to be reset. Before this procedure, the interchange of CAN messages via DMU is not possible and the Virtual Buffers may not be accessed.

48.2.2 Memory map

The DMU provides a memory mapped access to the embedded M_CAN with optional TSU at the first 512 addresses. The following addresses are for the DMU, the indirect addressing of the M_CAN head elements and the configuration registers of the DMU. Data may be accessed 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

Table 556. DMU memory map

Address offset	Name	Reset	Access	Location
0x0000–0x00FF	M_CAN address range	Refer to *Chapter 47: Controller area network (M_CAN)*		
0x0160–0x01FF	TSU Extension address range	Refer to *Chapter 49: Timestamping unit (TSU)*		
0x0200–0x0244	DMU TX Element (18)	MRAM(1)	W(2)	—
0x0248–0x027C	Reserved	—	—	—
0x0280–0x02C4	DMU RX0 Element (18)	MRAM(1)	R(2)	—
0x02C8–0x02FC	Reserved	—	—	—
0x0300–0x0344	DMU RX1 Element (18)	MRAM(1)	R(2)	—
0x0348–0x037C	Reserved	—	—	—
0x0380–0x0388	DMU TX Event Element (3)	MRAM(1)	R(2)	—

Table 556. DMU memory map (continued)

Address offset	Name	Reset	Access	Location
0x038C–0x03BF	Reserved	—	—	—
0x03C0	DMU Core Release (DMUCR)	0x10081114	R	*Section 48.2.3.1*
0x03C4	DMU Internals (DMUI)	0x00070000	RW	*Section 48.2.3.2*
0x03C8	DMU Queuing Counter (DMUQC)	0x00000000	RWC	*Section 48.2.3.3*
0x03CC	DMU Interrupt Register (DMUIR)	0x00000000	RC1	*Section 48.2.3.4*
0x03D0	DMU Interrupt Enable (DMUIE)	0x00000000	RW	*Section 48.2.3.5*
0x03D4	DMU Configuration (DMUC)	0x00000000	RP	*Section 48.2.3.6*
0x03D8–0x03FF	Reserved	—	—	—
0x0400–0x0444	DMU TX Element Debug (18)	MRAM(1)	R	—
0x0448–0x047F	Reserved	—	—	—
0x0480–0x04C4	DMU RX0 Element Debug (18)	MRAM(1)	R	—
0x04C8–0x04FF	Reserved	—	—	—
0x0500–0x0544	DMU RX1 Element Debug (18)	MRAM(1)	R	—
0x0548–0x057F	Reserved	—	—	—
0x0580–0x0584	DMU TX Event Element Debug (2)	MRAM(1)	R	—
0x0588–0x05FF	Reserved	—	—	—
0x0600–0x07FF	Reserved	—	—	—

1. Indirect access to MRAM by DMU. MRAM not initialized by HW reset.
2. Access controlled by DMU Element handler.

48.2.3 Registers

48.2.3.1 DMU Core Release Register (DMUCR)

Core release for coding of revisions (the coding of revisions depends on the module version used in the device).

Offset: 0x03C0 Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MON								DAY							
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. Refer to *Table 556: DMU memory map* for the device reset value of this register.

Figure 820. DMU Core Release Register (DMUCR)

Table 557. DMUCR field descriptions

Field	Description
31:28 REL	Core Release One digit, BCD-coded
27:24 STEP	Step of Core Release One digit, BCD-coded
23:20 SUBSTEP	Sub-step of Core Release One digit, BCD-coded
19:16 YEAR	Time Stamp Year One digit, BCD-coded. This field is set by generic parameter on DMU synthesis.
15:8 MON	Time Stamp Month Two digits, BCD-coded. This field is set by generic parameter on DMU synthesis.
7:0 DAY	Time Stamp Day Two digits, BCD-coded. This field is set by generic parameter on DMU synthesis.

48.2.3.2 DMU Internals register (DMUI)

This register provides internals of the DMU, which must be used for debugging only.

Offset: 0x03C4 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TXE	RX1	RX0	TX	0	EHS			DTXE	DTX1	DTX0	DTX	0	DEHS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENA	0	0	TFQPIP					0	0	0	0	TXER	RX1R	RX0R	TXR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 821. DMU Internals register (DMUI)

Table 558. DMUI field descriptions

Field	Description
31 TXE	Actual DMU Element Service TXE, RX1, RX0 and TX flags show which DMU Virtual Buffer is currently served, refer to *Figure 818: Address map*. Only one Virtual Buffer can be served at one time. The respective bit is set by accessing the service's Start Address. The bit is reset when service finishes after entering Element Handler State wait4sa (queuing successful) or recovery (queuing failure). 0 DMU Virtual Buffer is currently not served 1 DMU Virtual Buffer is currently served
30 RX1	Actual DMU Element Service TXE, RX1, RX0 and TX flags show which DMU Virtual Buffer is currently served, refer to *Figure 818: Address map*. Only one Virtual Buffer can be served at one time. The respective bit is set by accessing the service's Start Address. The bit is reset when service finishes after entering Element Handler State wait4sa (queuing successful) or recovery (queuing failure). 0 DMU Virtual Buffer is currently not served 1 DMU Virtual Buffer is currently served
29 RX0	Actual DMU Element Service TXE, RX1, RX0 and TX flags show which DMU Virtual Buffer is currently served, refer to *Figure 818: Address map*. Only one Virtual Buffer can be served at one time. The respective bit is set by accessing the service's Start Address. The bit is reset when service finishes after entering Element Handler State wait4sa (queuing successful) or recovery (queuing failure). 0 DMU Virtual Buffer is currently not served 1 DMU Virtual Buffer is currently served
28 TX	Actual DMU Element Service TXE, RX1, RX0 and TX flags show which DMU Virtual Buffer is currently served, refer to *Figure 818: Address map*. Only one Virtual Buffer can be served at one time. The respective bit is set by accessing the service's Start Address. The bit is reset when service finishes after entering Element Handler State wait4sa (queuing successful) or recovery (queuing failure). 0 DMU Virtual Buffer is currently not served 1 DMU Virtual Buffer is currently served
26:24 EHS	Element Handler State Current state of the DMU Element Handler FSM, which controls the interaction with all DMU Elements, that are TX, RX0, RX1, TXE. 000 (wait4cce) wait for bit M_CAN:CCCR.CCE getting zero 001 (wait4sa) wait for Start Address 010 (wait4ta) wait for Trigger Address 011 (transfer) wait for transfer of Element word 100 (ack2mcan) acknowledge to M_CAN 101 (recovery) exception recovery 110 Reserved 111 Reserved

Table 558. DMUI field descriptions (continued)

Field	Description
23 DTXE	Detect DMU Element Service DTXE, DRX1, DRX0, DTX define the DMU Element Services used for debugging. Several services may be selected at one time. Interrupt flag DMUIR.DT gets active, when DMU Element Service and Element Handler State matches. 0 Queuing of DMU Element does not activate interrupt flag 1 Queuing of DMU Element activates interrupt flag when DMUI.EHS = DMUI.DEHS
22 DTX1	Detect DMU Element Service DTXE, DRX1, DRX0, DTX define the DMU Element Services used for debugging. Several services may be selected at one time. Interrupt flag DMUIR.DT gets active, when DMU Element Service and Element Handler State matches. 0 Queuing of DMU Element does not activate interrupt flag 1 Queuing of DMU Element activates interrupt flag when DMUI.EHS = DMUI.DEHS
21 DTX0	Detect DMU Element Service DTXE, DRX1, DRX0, DTX define the DMU Element Services used for debugging. Several services may be selected at one time. Interrupt flag DMUIR.DT gets active, when DMU Element Service and Element Handler State matches. 0 Queuing of DMU Element does not activate interrupt flag 1 Queuing of DMU Element activates interrupt flag when DMUI.EHS = DMUI.DEHS
20 DTX	Detect DMU Element Service DTXE, DRX1, DRX0, DTX define the DMU Element Services used for debugging. Several services may be selected at one time. Interrupt flag DMUIR.DT gets active, when DMU Element Service and Element Handler State matches. 0 Queuing of DMU Element does not activate interrupt flag 1 Queuing of DMU Element activates interrupt flag when DMUI.EHS = DMUI.DEHS
18:16 DEHS	Detect Element Handler State Define the DMU Element Handler state used for debugging. When state of the DMU Element Handler FSM reaches that state, and DMU Element Service was chosen for debugging (see bits 23:20) the interrupt flag DMUIR.DT is activated. Reset value 7 = unused state. State encoding described in field description DMUI.EHS above.
15 ENA	DMU is enabled This flag shows the state of the DMU, enabled or disabled, depending on CCE mode of M_CAN, refer to *Section 48.3.6: M_CAN in CCE mode*. 0 DMU is disabled 1 DMU is enabled and can process DMA data
12:8 TFQPIP	TX FIFO/Queue Put Index Previous Put index, used by DMU Element Handler for previous enqueuing of a Tx Element. The debug section of Tx Element uses the previous put index for address redirection, to have access to the previously enqueued data.

Table 558. DMUI field descriptions (continued)

Field	Description
3 TXER	TX Event Service Request line of DMU Logical value of the output signal dmu_txer, which requests a service from the external DMA to dequeue Tx event information with DMU TX Event Element. 0 No TX Event DMA service requested 1 TX Event DMA Service requested
2 RX1R	RX1 Service Request line of DMU Logical value of the output Signal dmu_rx1r, which requests a service from the external DMA to dequeue a CAN message with DMU RX1 Element. 0 No RX1 DMA service requested 1 RX1 DMA Service requested
1 RX0R	RX0 Service Request line of DMU Logical value of the output signal dmu_rx0r, which requests a service from the external DMA to dequeue a CAN message with DMU RX0 Element. 0 No RX0 DMA service requested 1 RX0 DMA Service requested
0 TXR	TX Service Request line of DMU Logical value of the output signal dmu_txr, which requests a service from the external DMA to enqueue a CAN message with DMU TX Element. 0 No TX DMA service requested 1 TX DMA Service requested

48.2.3.3 DMU Queuing Counter register (DMUQC)

Every successful queuing is counted, for each DMU Element separately. The counters wrap around at the end-value, that is modulo 256. All counters can be reset by a write access, write data is Don't care.

Offset: 0x03C8 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TXEEDC								RX1EDC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX0EDC								TXEEC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 822. DMU Queuing Counter register (DMUQC)

Table 559. DMUQC field descriptions

Field	Description
31:24 TXEEDC	TX Event Element Dequeuing Counter Increments on each successful dequeuing of a CAN TX Event information via DMU TX Event Element.
23:16 RX1EDC	RX1 Element Dequeuing Counter Increments on each successful dequeuing of a CAN RX message via DMU RX1 Element.
15:8 RX0EDC	RX0 Element Dequeuing Counter Increments on each successful dequeuing of a CAN RX message via DMU RX0 Element.
7:0 TXEEC	TX Element Enqueuing Counter Increments on each successful enqueuing of a CAN TX message via DMU TX Element.

48.2.3.4 DMU Interrupt Register (DMUIR)

This register provides information about occurrence of several events inside the DMU. A flag is set when the related condition is detected (edge-sensitive). The flags remain set until the Host (CPU) clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset clears the register. The configuration of DMUIE controls whether the DMU signals a level based interrupt on the interrupt line dmu_int.

Offset: 0x03CC Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	IAC	DT	TXEED	TXEEIW	TXEEIAS	TXEEID	TXEENSA	0	0	RX1EIO	RX1ED	RX1EIW	RX1EIAS	RX1EID	RX1ENSA
W		w1c		w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BEU	BEC	RX0EIO	RX0ED	RX0EIW	RX0EIAS	RX0EID	RX0ENSA	0	TXEE	TXEIR	TXEWATA	TXEIDLC	TXEIAS	TXEIE	TXENSA
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 823. DMU Interrupt Register (DMUIR)

Table 560. DMUIR field descriptions

Field	Description
30 IAC	Illegal Access while in Configuration mode As long as the associated M_CAN is in CCE mode (M_CAN:CCCR.CCE = 1), the DMU is disabled and the Element Handler FSM is hold in state wait4cce, enqueuing and dequeuing is not possible, see *Section 48.3.6: M_CAN in CCE mode*. Any access to the DMU Virtual Buffers (TX, RX0, RX1, TX Event) sets the flag. 0 No Illegal Access while CCE mode 1 Illegal Access while CCE mode
29 DT	Debug Trigger In register DMUI the user can define a trigger condition for debugging, that is when a requested state of the DMU Element Handler is reached combined with the requested DMU Element Service. refer to register DMUI, bits 31:16. 0 Debug point not reached 1 Debug point reached
28 TXEED	TX Event Element Dequeued A successful dequeuing of a TX Event Element by the DMU sets this flag. 0 No TX Event Element dequeued 1 TX Event Element successfully dequeued
27 TXEEIW	TX Event Element Illegal Write All write accesses to the DMU TX Event Element are illegal. 0 No write access detected 1 Illegal write access to DMU TX Event Element detected, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
26 TXEEIAS	TX Event Element Illegal Access Sequence When the Start Address of the TX Event Element was accessed, the DMU expects the transfer of the TX Event Element. These read accesses to DMU Section have to be strictly linear to ascending and consecutive addresses until reaching the Trigger Address. This flag gets also active, when accessing another Virtual Buffer, before TX Event Element Service has been completed. 0 No illegal addressing sequence detected 1 Accesses are not strictly linear to ascending and consecutive addresses, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
25 TXEEID	TX Event Element Illegal Dequeuing A dequeuing of a TX Event Element with the DMU must only be started (reading from the Start Address of the TX Event Virtual Buffer) if this is requested by DMU Request signal dmu_txer. 0 No illegal dequeuing 1 Start of dequeuing without request detected, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*

Table 560. DMUIR field descriptions (continued)

Field	Description
24 TXEENSA	TX Event Element Not Start Address An interaction with the TX Event Element Section of the DMU always has to start with a read access to the start address. This detector is only active when DMU is idle and thus awaits a start of a new service by an access to one of the Virtual Buffer's start addresses. 0 No illegal read access 1 Read from TX Event Element begins without using start address, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
21 RX1EIO	RX1 Element Illegal Overwrite by timestamp The configuration of the M_CAN Data Field Size must ensure, that the last word of a Rx Element is not used by SYNC message itself, in order to avoid DMU-internal overwrite with timestamp, refer to *Section 48.3.5.1: Timestamp at DMU RX element*. 0 No illegal overwrite detected 1 DMU has internally overwritten the last element word of a SYNC message
20 RX1ED	RX1 Element Dequeued A successful dequeuing of a Rx message with the DMU RX1 Element sets this flag 0 No Rx message dequeued 1 Rx message successfully dequeued
19 RX1EIW	RX1 Element Illegal Write All write access to the DMU RX1 Element are illegal. 0 No write access detected 1 Illegal write access to DMU RX1 Element detected, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
18 RX1EIAS	RX1 Element Illegal Access Sequence When the start address of the RX1 Element was accessed, the DMU expects the transfer of the CAN message. These read accesses to DMU RX1 Element have to be strictly linear to ascending and consecutive addresses until reaching the Trigger Address. This flag gets also active, when accessing another Virtual Buffer, before RX1 Element service has been completed. 0 No illegal addressing sequence detected 1 Accesses are not strictly linear to ascending and consecutive addresses, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
17 RX1EID	RX1 Element Illegal Dequeuing A dequeuing of a Rx message with the Rx section of the DMU must only be started (reading from the start address of the RX1 Virtual Buffer) if this is requested by DMU Request signal dmu_rx1r. 0 No illegal dequeuing 1 Start of dequeuing without request detected, exception recovery started,

Table 560. DMUIR field descriptions (continued)

Field	Description
16 RX1ENSA	RX1 Element Not Start Address An interaction with the RX1 section of the DMU always has to start with a read access to the start address. This detector is only active when DMU is idle and thus awaits a start of a new service by an access to one of the Virtual Buffer's start addresses. 0 No illegal read access 1 Read from RX1 Element begins without using start address, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
15 BEU	Bit Error Uncorrected On read accesses to the MRAM via DMU, an optional external parity/ECC logic attached to the MRAM can detect bit errors. 0 No bit error detected when reading from Message RAM 1 Uncorrected bit error detected
14 BEC	Bit Error Corrected On read accesses to the MRAM via DMU, an optional external ECC logic attached to the MRAM can detect and correct bit errors. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected
13 RX0EIO	RX0 Element Illegal Overwrite by timestamp The configuration of the M_CAN Data Field Size must ensure, that the last word of a Rx Element is not used by SYNC message itself, in order to avoid DMU-internal overwrite with timestamp, refer to *Section 48.3.5.1: Timestamp at DMU RX element*. 0 No illegal overwrite detected 1 DMU has internally overwritten the last element word of a SYNC message
12 RX0ED	RX0 Element Dequeued A successful dequeuing of a Rx message with the DMU RX0 Element sets this flag. 0 No Rx message dequeued 1 Rx message successfully dequeued
11 RX0EIW	RX0 Element Illegal Write All write access to the DMU RX0 Element are illegal. 0 No write access detected 1 Illegal write access to DMU RX1 Element detected, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
10 RX0EIAS	RX0 Element Illegal Access Sequence When the start address of the RX0 Element was accessed, the DMU expects the transfer of the CAN message. These read accesses to DMU RX0 Element have to be strictly linear to ascending and consecutive addresses until reaching the Trigger Address. This flag gets also active, when accessing another Virtual Buffer, before RX0 Element service has been completed. 0 No illegal addressing sequence detected 1 Accesses are not strictly linear to ascending and consecutive addresses, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*

Table 560. DMUIR field descriptions (continued)

Field	Description
9 RX0EID	RX0 Element Illegal Dequeuing A dequeuing of a Rx message with the Rx section of the DMU must only be started (reading from the start address of the RX0 Virtual Buffer) if this is requested by DMU Request signal dmu_rx0r. 0 No illegal dequeuing 1 Start of dequeuing without request detected, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
8 RX0ENSA	RX0 Element Not Start Address An interaction with the RX0 section of the DMU always has to start with a read access to the start address. This detector is only active when DMU is idle and thus awaits a start of a new service by an access to one of the Virtual Buffer's start addresses. 0 No illegal read access 1 Read from RX0 Element begins without using start address, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
6 TXEE	TX Element Enqueued A successful enqueuing of a Tx message with the DMU TX Element section sets this flag. 0 No Tx message enqueued 1 Tx message successfully enqueued
5 TXEIR	TX Element Illegal Read All read accesses to the DMU TX Element section are illegal and return undefined read data. 0 No read access 1 Illegal read access to DMU TX Element section detected, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
4 TXEWATA	TX Element Write After Trigger Address The DMU expects that only the active words of a TX Element are transferred, refer to *Section 48.3.1.1: Enqueue a message*. In some applications, the message transfer with dynamic length, controlled by DLC is not applicable. In such a case, the DMA controller may transfer (write) the maximum size Tx Buffer element, independent of the actual length of the contained message. The DMU tolerates this by discarding these accesses and setting this flag. This exception handling is finished with accessing an address other than between Trigger Address and last address of TX Element section or on occurrence of an access error, which normally aborts a pending enqueuing. 0 No write after Trigger Address 1 Write after Trigger address detected

Table 560. DMUIR field descriptions (continued)

Field	Description
3 TXEIDLC	TX Element Illegal DLC In the case that the DLC written to TX Element indicates a message that is larger than the configuration of M_CAN's Tx Buffer element size (M_CAN TXESC.TBDS), the DMU aborts the enqueuing. Example: Tx Buffer element size is set to 8 byte data fields (M_CAN TXESC.TBDS = 0). The DLC in T1 written to DMU TX Element indicates a CAN message with 64 byte data field. 0 No illegal DLC detected 1 DLC exceeds Tx Buffer element size of M_CAN, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
2 TXEIAS	TX Element Illegal Access Sequence By accessing the start address of the TX Element, the DMU awaits the transfer of the CAN message. These write accesses to DMU TX Element have to be strictly linear to ascending and consecutive addresses until reaching the last word of the CAN message. This flag gets also active, when accessing another Virtual Buffer, before TX Element service has been completed. 0 No illegal DLC detected 1 DLC exceeds Tx Buffer element size of M_CAN, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
1 TXEIE	TX Element Illegal Enqueuing An enqueueing of a Tx message with the TX Section of the DMU must only be started (writing to the start address of the TX Section) if this is requested by DMU Request signal dmu_txr. 0 No illegal enqueuing 1 Start of enqueuing without request detected, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*
0 TXENSA	TX Element Not Start Address An interaction with the TX Section of the DMU always has to start with a write access to the start address. This detector is only active when DMU is idle and thus awaits a start of a new service by an access to one of the Virtual Buffer's start addresses. 0 No illegal write access 1 Write to TX Element begins without using start address, exception recovery started, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*

48.2.3.5 DMU Interrupt Enable register (DMUIE)

The register controls whether the DMU signals an interrupt on interrupt line dmu_int. Thus, for each flag in the Interrupt register DMUIR this register provides a dedicated enable bit.

Offset: 0x03D0 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	IACE	DTE	TXEEDE	TXEEIWE	TXEEIASE	TXEEIDE	TXEENSAE	0	0	RX1EIOE	RX1EDE	RX1EIWE	RX1EIASE	RX1EIDE	RX1ENSAE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BEUE	BECE	RX0EIOE	RX0EDE	RX0EIWE	RX0EIASE	RX0EIDE	RX0ENSAE	0	TXEEE	TXEIRE	TXEWATAE	TXEIDLCE	TXEIASE	TXEIEE	TXENSAE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 824. DMU Interrupt Enable register (DMUIE)

Table 561. DMUIE field descriptions

Field	Description
30 IACE	Illegal Access while in Configuration mode Enable 0 Flag DMUIR.IAC does not activate the interrupt line dmu_int. 1 If DMUIR.IAC = 1 the interrupt line dmu_int is activated.
29 DTE	Debug Trigger Enable 0 Flag DMUIR.DT does not activate the interrupt line dmu_int. 1 If DMUIR.DT = 1 the interrupt line dmu_int is activated.
28 TXEEDE	TX Event Element Dequeued 0 Flag DMUIR.TXEED does not activate the interrupt line dmu_int. 1 If DMUIR.TXEED = 1 the interrupt line dmu_int is activated.
27 TXEEIWE	TX Event Element Illegal Write 0 Flag DMUIR.TXEEIW does not activate the interrupt line dmu_int. 1 If DMUIR.TXEEIW = 1 the interrupt line dmu_int is activated.
26 TXEEIASE	TX Event Element Illegal Access Sequence 0 Flag DMUIR.TXEEIAS does not activate the interrupt line dmu_int. 1 If DMUIR.TXEEIAS = 1 the interrupt line dmu_int is activated.
25 TXEEIDE	TX Event Element Illegal Dequeuing 0 Flag DMUIR.TXEEID does not activate the interrupt line dmu_int. 1 If DMUIR.TXEEID = 1 the interrupt line dmu_int is activated.
24 TXEENSAE	TX Event Element Not Start Address 0 Flag TXEENSA.TXEENSA does not activate the interrupt line dmu_int. 1 If TXEENSA.TXEENSA = 1 the interrupt line dmu_int is activated.
21 RX1EIOE	RX1 Element Illegal Overwrite by timestamp Enable 0 Flag TXEENSA.RX1EIO does not activate the interrupt line dmu_int. 1 If TXEENSA.RX1EIO = 1 the interrupt line dmu_int is activated.
20 RX1EDE	RX1 Element Dequeued Enable 0 Flag TXEENSA.RX1ED does not activate the interrupt line dmu_int. 1 If TXEENSA.RX1ED = 1 the interrupt line dmu_int is activated.

Table 561. DMUIE field descriptions (continued)

Field	Description
19 RX1EIWE	RX1 Element Illegal Write Enable 0 Flag TXEENSA.RX1EIW does not activate the interrupt line dmu_int. 1 If TXEENSA.RX1EIW = 1 the interrupt line dmu_int is activated.
18 RX1EIASE	RX1 Element Illegal Access Sequence Enable 0 Flag TXEENSA.RX1EIAS does not activate the interrupt line dmu_int. 1 If TXEENSA.RX1EIAS = 1 the interrupt line dmu_int is activated.
17 RX1EIDE	RX1 Element Illegal Dequeuing Enable 0 Flag TXEENSA.RX1EID does not activate the interrupt line dmu_int. 1 If TXEENSA.RX1EID = 1 the interrupt line dmu_int is activated.
16 RX1ENSAE	RX1 Element Not Start Address Enable 0 Flag TXEENSA.RX1ENSA does not activate the interrupt line dmu_int. 1 If TXEENSA.RX1ENSA = 1 the interrupt line dmu_int is activated.
15 BEUE	Bit Error Uncorrected Enable 0 Flag TXEENSA.BEU does not activate the interrupt line dmu_int. 1 If TXEENSA.BEU = 1 the interrupt line dmu_int is activated.
14 BECE	Bit Error Corrected Enable 0 Flag TXEENSA.BEC does not activate the interrupt line dmu_int. 1 If TXEENSA.BEC = 1 the interrupt line dmu_int is activated.
13 RX0EIOE	RX0 Element Illegal Overwrite by timestamp Enable 0 Flag TXEENSA.RX0EIO does not activate the interrupt line dmu_int. 1 If TXEENSA.RX0EIO = 1 the interrupt line dmu_int is activated.
12 RX0EDE	RX0 Element Dequeued Enabled 0 Flag TXEENSA.RX0ED does not activate the interrupt line dmu_int. 1 If TXEENSA.RX0ED = 1 the interrupt line dmu_int is activated.
11 RX0EIWE	RX0 Element Illegal Write Enabled 0 Flag TXEENSA.RX0EIW does not activate the interrupt line dmu_int. 1 If TXEENSA.RX0EIW = 1 the interrupt line dmu_int is activated.
10 RX0EIASE	RX0 Element Illegal Access Sequence Enabled 0 Flag TXEENSA.RX0EIAS does not activate the interrupt line dmu_int. 1 If TXEENSA.RX0EIAS = 1 the interrupt line dmu_int is activated.
9 RX0EIDE	RX0 Element Illegal Dequeuing Enabled 0 Flag TXEENSA.RX0EID does not activate the interrupt line dmu_int. 1 If TXEENSA.RX0EID = 1 the interrupt line dmu_int is activated.
8 RX0ENSAE	RX0 Element Not Start Address Enabled 0 Flag TXEENSA.RX0ENSA does not activate the interrupt line dmu_int. 1 If TXEENSA.RX0ENSA = 1 the interrupt line dmu_int is activated.
6 TXEEE	TX Element Enqueued Enable 0 Flag TXEENSA.TXEE does not activate the interrupt line dmu_int. 1 If TXEENSA.TXEE = 1 the interrupt line dmu_int is activated.
5 TXEIRE	TX Element Illegal Read Enable 0 Flag TXEENSA.TXEIR does not activate the interrupt line dmu_int. 1 If TXEENSA.TXEIR = 1 the interrupt line dmu_int is activated.

Table 561. DMUIE field descriptions (continued)

Field	Description
4 TXEWATAE	TX Element Write After Trigger Address Enable 0 Flag TXEENSA.TXEWATA does not activate the interrupt line dmu_int. 1 If TXEENSA.TXEWATA = 1 the interrupt line dmu_int is activated.
3 TXEIDLCE	TX Element Illegal DLC Enable 0 Flag TXEENSA.TXEIDLC does not activate the interrupt line dmu_int. 1 If TXEENSA.TXEIDLC = 1 the interrupt line dmu_int is activated.
2 TXEIASE	TX Element Illegal Access Sequence Enable 0 Flag TXEENSA.TXEIAS does not activate the interrupt line dmu_int. 1 If TXEENSA.TXEIAS = 1 the interrupt line dmu_int is activated.
1 TXEIEE	TX Element Illegal Enqueuing Enable 0 Flag TXEENSA.TXEIE does not activate the interrupt line dmu_int. 1 If TXEENSA.TXEIE = 1 the interrupt line dmu_int is activated.
0 TXENSAE	TX Element Not Start Address Enable 0 Flag DMUIR.TXENSA does not activate the interrupt line dmu_int. 1 If DMUIR.TXENSA = 1 the interrupt line dmu_int is activated.

48.2.3.6 DMU Configuration register (DMUC)

This is the configuration register of the DMU.

Offset: 0x03D4 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 825. DMU Interrupt Enable register (DMUC)

Table 562. DMUC field descriptions

Field	Description
0 TTS	Transfer Timestamp The DMU supports the transfer of TSU timestamps via DMU Virtual Buffer for RX0, RX1 and TX Event Elements, refer to *Section 48.3.4: Exception recovery at DMU virtual buffer*. This flag is kept to 0 if M_CAN CCCR.UTSU = 0, see M_CAN chapter. 0 No timestamp is transferred via DMU Virtual Buffer. 1 Timestamp of message is transferred from TSU via DMU Virtual Buffer.

48.3 Functional description

48.3.1 DMU TX element (0x200–0x244)

This section describes the TX Virtual Buffer of the DMU. It allows to enqueue messages to the Tx FIFO/Queue of the M_CAN in a simplified way. Both queue types of the M_CAN are supported: The so called “Tx Queue” and the so called “Tx FIFO”. The configuration of the M_CAN Tx FIFO/Queue has to be done exactly in the same way as without using DMU, since the DMU provides just another access path to the M_CAN's Tx FIFO/Queue elements and does not supersede them.

48.3.1.1 Enqueue a message

An enqueuing of a Tx message via the TX Element section of the DMU must only be started if this is requested by DMU Request signal dmu_txr.

The CAN message format of the DMU TX Element is the format described in *Section 47.2.4.2: Tx buffer element* from M_CAN chapter. To enqueue a Tx message, a strict write access sequence has to be processed. It has to start by writing the T0 word of the CAN message to the first address of the TX Virtual Buffer (0x200), which is called the start address. This triggers the DMU to expect a CAN Tx message transfer. The CAN message has to be transferred by linear addressing to ascending and consecutive addresses, word by word. The transfer completes, by writing the last element word Tn (with n = {1, 2, 3, 4, 5, 6, 7, 9, 13, 17}) which is called is the Trigger Address. It is the last required write access to the TX Element Section for a particular CAN message, which triggers the scheduling of that TX message element inside the M_CAN.

The DMU derives the Trigger Address from the RTR bit in T0, the FDF bit in T1, and the Data Length Code (DLC) in T1. This means the DMU derives the Trigger Address dynamically for each message, depending on the individual payload to be transferred. In case of a CAN message with DLC = 4 the Trigger Address is T2. After completion, normally the next expected access to the DMU Element sections is a start address. The DMU TX Element section is also compatible to DMA with fixed transfer length (independent of payload), see flag DMUIR.TXEWATA.

Status flags in the Interrupt Register DMUIR provide detailed information, when one of the access rules is violated. It is highly recommended, to carefully check all DMUIR.TXExxx status flags during runtime for unintended activations.

The following flags must never get active during normal operation:

- DMUIR.TXENSA:TX Element Not Start Address
- DMUIR.TXEIE:TX Element Illegal Enqueuing
- DMUIR.TXEIAS:TX Element Illegal Access Sequence
- DMUIR.TXEIDLC:TX Element Illegal DLC
- DMUIR.TXEIR:TX Element Illegal Read

48.3.2 DMU RX element (0x280–0x2C4 and 0x300–0x344)

This section describes the DMU RX0 Element and DMU RX1 Element handling. The two Elements behave identically, therefore the expression DMU RX Element is used, which describes both.

The DMU RX Element allows dequeuing of messages from Rx FIFO 0,1 of the M_CAN in a simplified way. The DMU RX Element behaves similarly to the previously described TX

Element (refer to *Section 48.3.1: DMU TX element (0x200–0x244)*). The main differences are the transfer direction (MRAM ' CRAM) and that the transfer size is constant (according to M_CAN RXESC.FnDS) and does not dynamically change by DLC of a given message.

48.3.2.1 Dequeuing a message

Dequeuing of an Rx message with the DMU RX Element must only be started, if this is requested by DMU Request signals dmu_rx0r respective dmu_rx1r.

The message format of the DMU RX Element is the format, described in M_CAN chapter, section Rx Buffer and FIFO Element. To dequeue an Rx message, a strict read access sequence has to be processed. It has to start by reading the R0 word of the CAN message from the first address of the DMU RX Element (0x280 resp. 0x300), which is called the start address. This triggers the DMU to expect a CAN Rx message transfer. The CAN message has to be transferred by linear addressing to ascending and consecutive addresses, word by word. The transfer completes, by reading the Trigger Address Rn, with n = {3, 4, 5, 6, 7, 9, 13, 17}.

The Trigger Address is derived from the configuration of the dedicated M_CAN Rx FIFO n Element Size. For example if M_CAN RXESC.F0DS = 0 and M_CAN RXESC.F1DS = 7, the Trigger Address of DMU RX 0 Element is 0x28C (Rx FIFO element word R3) and Trigger Address of DMU RX 1 Element is 0x344 (Rx FIFO element word R17). After completion, the next expected access to the DMU Element sections is a start address.

Status flags in the Interrupt Register DMUIR provide detailed information, when one of the access rules is violated. It is highly recommended, to carefully check all DMUIR.RXnExxx status flags during runtime for unexpected activations.

The following flags must never get active during normal operation:

- DMUIR.RX0,1EIO: RX Element Illegal Overwrite
- DMUIR.RX0,1EIW: RX Element Illegal Write
- DMUIR.RX0,1EIAS: RX Element Illegal Access Sequence
- DMUIR.RX0,1EID: RX Element Illegal Dequeuing
- DMUIR.RX0,1ENSA: RX Element Not Start Address

48.3.3 TX event element (0x380–0x388)

This section describes the TX Event Element of the DMU, which allows to dequeue Tx events from the Tx Event FIFO of the M_CAN in a simplified way.

The Tx Event FIFO of the M_CAN provides information about transmitted CAN messages (see *Section 47.2.4.3: Tx event FIFO element* from M_CAN chapter). Like with the Rx FIFOs, data has to be transferred from MRAM ' CRAM. Data handling is the same as with the DMU RX Element. The only difference is, that the Trigger Address is 0x384 when DMUC.TTS = '0' or 0x388 when DMUC.TTS = '1', which is related to the M_CAN Tx Event FIFO word E1 respective TSU timestamp.

Status flags of the Interrupt Register DMUIR provide detailed information when one of the access rules is violated. It is highly recommended, to carefully check all DMUIR.TXEExxx status flags during runtime for unintended activations.

The following flags must never get active during normal operation:

- DMUIR.TXEEIW: TX Event Element Illegal Write
- DMUIR.TXEEIAS: TX Event Element Illegal Access Sequence
- DMUIR.TXEEID: TX Event Element Illegal Dequeuing
- DMUIR.TXEENSA: TX Event Element Not Start Address

48.3.4 Exception recovery at DMU virtual buffer

If the DMU detects an illegal access to the DMU Virtual Buffers, it starts an exception recovery in two steps.

The first step is to immediately abort an ongoing service, that is a partly transferred Tx message is discarded, that is not scheduled at M_CAN, to avoid corrupted messages on the CAN bus, respective a partly transferred Rx message regarded as not transferred.

The second step is to wait for the next access to a start address of one of the DMU Virtual Buffers. Accesses to Virtual Buffer addresses except one of the four Start Addresses are ignored and do not activate further interrupt flags, so the active status flag in the DMUIR register shows the first occurred access error.

48.3.5 Transferring timestamps from TSU

The M_CAN may be equipped with a Timestamping Unit (TSU), which is an add-on to the M_CAN IP-module for hardware timestamping support according to CiA 603 [3]. The TSU module can be used by setting M_CAN CCCR.UTSU = '1'. Received Sync messages have a pointer to the related TSU timestamp register. The register holds the 32-bit timestamp captured by the TSU, see M_CAN chapter.

Note: DMUC.TTS can only be activated if the M_CAN has attached a TSU and the TSU is enabled by M_CAN CCCR.UTSU = '1'.

Flag TSC (Time Stamp Captured) is located in the Sync message's element word R1B/E1B stored in the M_CAN's Message RAM, see M_CAN chapter.

Debugging access does not support redirection to TSU timestamp. Instead, undefined read data is returned. To check the timestamps stored in the TSU, they have to be addressed directly.

48.3.5.1 Timestamp at DMU RX element

The DMU is able to transfer the TSU timestamps, by replacing the element word Rn at the Trigger Address with the referenced timestamp of the TSU. Therefore, the Rx FIFO Data Field Size must be at least 4 bytes larger than the data field of the Sync messages, in order to avoid replacement of payload by the timestamp. For configuration of the M_CAN Data Field Size, see M_CAN chapter, register M_CAN RXESC (Rx Buffer / Rx FIFO Element Size Configuration).

The replacement of element word Rn is only done if both following conditions are true:

- Timestamp transport in DMU is enabled by DMUC.TTS = 1
- Flag R1B.TSC = 1

48.3.5.2 Timestamp at DMU TX event element

When the transport of timestamps via DMU is enabled by DMUC.TTS = 1, the Trigger Address of the TX Event Element is shifted by one element word, that is from E1 to E1+1.

Through this, the DMU has space for the optional transfer of the TSU timestamp at last element word.

If in TTS mode and the flag TSC (Time Stamp Captured) in the Tx Event FIFO element word E1B is set, the last word contains the TSU timestamp, otherwise the value of the last word is undefined.

- Enabling DMU Timestamp transport by DMUC.TTS = 1
  DMU TX Event Element size switches from 2 to 3 words
- If condition above is true and flag E1B.TSC = 1
  DMU provides TSU Timestamp at last element word

48.3.6 M_CAN in CCE mode

After hardware reset the DMU is disabled. If the DMU is disabled, it waits for a transition from 1 to 0 of bit M_CAN CCCR.CCE, which finishes M_CAN's CCE mode. A new activation of CCE mode disables again the DMU.

If the DMU is disabled it behaves in the following way:

- Access to the DMU Element sections (TX, RX0, RX1, TX Event) is not allowed. If accessed anyhow, a read returns undefined read data respective a write is discarded, and flag DMUIR.IAC is set.
- The DMU Element Handler set to reset state, for example a pending transaction is aborted immediately. Therefore it is highly recommended, to shut down DMA services for the DMU before entering CCE mode.
- All DMU Service Request signals are set to zero.

48.3.7 DMU service request signals

The DMU provides signals, to trigger an external DMA Controller for a service, to transfer a M_CAN FIFO element from CRAM to MRAM via the DMU TX Element, or vice versa from MRAM to CRAM via the DMU RX Elements respective DMU TX Event Element. Refer also to *Section 48.1.2: Message handling with DMU*.

48.3.7.1 Timing of service request signals

The DMU Service Request signals get active, when the following conditions are met:

- M_CAN CCCR.CCE = ‘0’
- Service of Virtual Buffer is not ongoing
- For DMU TX Virtual Buffer: M_CAN Tx FIFO/Tx Queue is not full
- For other DMU Virtual Buffers: M_CAN Rx FIFO/Tx Event FIFO is not empty

The external DMA Controller must start a service only if requested by DMU Service Request signals. When the DMA Controller starts the service by accessing the Start Address of a DMU Element, the Service Request signal of the addressed Element is de-asserted.

The DMA Controller completes the service by accessing the Trigger Address. Now the DMU starts internal follow-up interactions with M_CAN, for example acknowledge of the element access. On completion of these tasks, the Service Request signal of the dedicated element is re-enabled and becomes active, as soon as the service is requested again for example due to M_CAN FIFO status.

48.3.7.2 Unused service request signals

If an application does not use the full DMU support, for example because some M_CAN FIFOs are accessed directly by CPU, the unused DMU Service Request signals have to be ignored by external DMA. Disabling by DMU is not supported.

48.3.8 Software debugging support

The DMU provides three different kinds of debugging support function:

- Indirect access to Buffer Elements by DMU Debug Virtual Buffers
- Programmable trigger condition for Element Handler, with interrupt
- Direct access to MRAM and M_CAN

48.3.8.1 DMU debug sections

The following DMU Debug Sections provide indirect access to previous or current DMU Virtual Buffers.

Table 563. Debug sections

Section	Addresses	Head element
TX Element Debug	0x400–0x444	previous
RX 0 Element Debug	0x480–0x4C4	current
RX 1 Element Debug	0x500–0x544	current
TX Event Element Debug	0x580–0x584	current

These sections are intended to be used by a debugging tool (for example connected via JTAG), that is to check the CAN message of RX0 Element by using the same redirection mechanism like the DMU Element Handler.

Only read access is supported, write accesses are discarded. The accesses do not influence the DMU Element Handler, so a read is possible at any time, even when a queuing is ongoing. Read access can be done randomly without constraints and no Start/Trigger Addresses exist.

For the TX Element, the previous address redirection of the DMU Element Handler is used, to have access to the previously enqueued data.

For RX0, RX1 and TX Event Elements, the debug path uses the current address redirection of the DMU Element Handler, to see element data that is dequeued next. The following table lists the registers for all put and get indexes:

Table 564. Registers for put/get indexes

Section	Register
TX Element Debug section	DMU DMUI.TFQPIP
TX Element (current put index)	M_CAN TXFQS.TFQPI
RX 0 Element Debug section	M_CAN RXF0S.F0GI
RX 1 Element Debug section	M_CAN RXF1S.F1GI
TX Event Element Debug section	M_CAN TXEFS.EFGI

The following constraints have to be regarded, using the debug feature:

- Debugging access does not support redirection to TSU timestamp. Instead, undefined read data is returned. Access the TSU directly, to check timestamps.
- When reading with debugger, while a DMU service is ongoing, the index of a DMU Virtual Buffer may change.

48.3.8.2 Programmable trigger

While debugging, it is often necessary to check data consistency of the DMU at defined internal states, called debug points. Therefore, such debug points may be selected by register DMUI in two steps (refer also to *Section 48.2.3.2: DMU Internals register (DMUI)*):

- Define one or multiple DMU Element Services of interest by bits DMUI.DTXE, DMUI.DRX1, DMUI.DRX0, DMUI.DTX
- Define target state of Element Handler FSM by DMUI.DEHS

When reaching the requested DMU state and service, the interrupt flag DMUIR.DT is set and triggers the CPU by interrupt to check data consistency of the DMU.

The following things have to be considered when using programmable trigger:

- In states wait4cce and recovery no Element Service is done at any time. When one of these states is selected as target state by DMUI.DEHS, the bits DMUI.DTXE, DMUI.DRX1, DMUI.DRX0, DMUI.DTX do not care, and entering the requested state activates interrupt flag DMUIR.DT.
- Interrupt flag DMUIR.DT is edge triggered, that is when the requested debug point starts matching to DMU internal state, flag gets active. The no-match to match transition can occur by programming a new debug point, as well by internal state transitions.

49 Timestamping unit (TSU)

49.1 Introduction

The Timestamping Unit (TSU) is an add-on to the M_CAN IP-module. Together with the M_CAN it supports hardware timestamping according to CiA 603.

49.2 Memory map and register descriptions

49.2.1 Memory map

The TSU is mapped into the M_CAN address space starting at address 0x0160. All registers are 32-bit registers.

Table 565. TSU memory map

Address offset	Register	Reset	Access	Location
0x0000–0x00FF	M_CAN address range	Refer to *Chapter 47: Controller area network (M_CAN)*		
0x0160	Core Release register (CREL)	0x10081114	R	*Section 49.2.2.1*
0x0164	Timestamp Configuration register (TSCFG)	0x00000000	RP	*Section 49.2.2.2*
0x0168	Timestamp Status 1 register (TSS1)	0x00000000	RWC	*Section 49.2.2.3*
0x016C	Timestamp Status 2 register (TSS2)	0x00006000	R	*Section 49.2.2.4*
0x0170 + n*0x4	Timestamp n register (TSn) for n =0 to 15	0x00000000	R	*Section 49.2.2.5*
0x01B0	Actual Timebase (ATB)	0x00000000	RC	*Section 49.2.2.6*
0x01B4–0x01FF	Reserved			
0x0200–0x05FF	DMU (DMU registers start at 0x03C0)	Refer to *Chapter 48: DMA interface unit (DMU)*		
0x0600–0x0FFF	Reserved			

49.2.2 Registers

49.2.2.1 Core Release register (CREL)

Core release for coding of revisions (the coding of revisions depends on the module version used in the device).

Offset: 0x0160 Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MON								DAY							
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. Refer to *Table 565: TSU memory map* for the device reset value of this register.

Figure 826. Core Release register (CREL)

Table 566. CREL field descriptions

Field	Description
31:28 REL	Core Release One digit, BCD-coded
27:24 STEP	Step of Core Release One digit, BCD-coded
23:20 SUBSTEP	Sub-step of Core Release One digit, BCD-coded
19:16 YEAR	Time Stamp Year One digit, BCD-coded. This field is set by generic parameter on DMU synthesis
15:8 MON	Time Stamp Month Two digits, BCD-coded. This field is set by generic parameter on DMU synthesis
7:0 DAY	Time Stamp Day Two digits, BCD-coded. This field is set by generic parameter on DMU synthesis

49.2.2.2 Timestamp Configuration register (TSCFG)

Offset: 0x0164 Access: Read protected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBPRE								0	0	0	0	0	SCP	TBCS	TSUE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 827. Timestamp Configuration register (TSCFG)

Table 567. TSCFG field descriptions

Field	Description
15:8 TBPRE	Timebase Prescaler (0x00 to 0xFF) The value by which the oscillator frequency is divided for generating the timebase counter clock. Valid values for the Timebase Prescaler are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Affects only the TSU internal timebase. When the internal timebase is excluded by synthesis, TBPRE[7:0] is fixed to 0x00, the Timestamp Prescaler is not used.
2 SCP	Select Capturing Position With generic internal_tb_sof_g = 0 or 1 this bit is not available, reading always returns 0. 0 Capture Timestamp at EOF 1 Capture Timestamp at SOF
1 TBCS	Timebase Counter Select When the internal timebase is excluded by synthesis, TBCS is fixed to 1. 0 Timestamp value captured from internal timebase counter, ATB.TB[31:0] is the internal timebase counter 1 Timestamp value captured from input tsu_tbin[31:0], ATB.TB[31:0] is tsu_tbin[31:0]
0 TSUE	Timestamp Unit Enable 0 0 TSU disabled 1 TSU enabled

49.2.2.3 Timestamp Status 1 register (TSS1)

Every successful queuing is counted, for each DMU Element separately. The counters wrap around at the end-value, that is modulo 256. All counters can be reset by a write access, write data is Don't care.

Offset: 0x0168 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TSL15	TSL14	TSL13	TSL12	TSL11	TSL10	TSL9	TSL8	TSL7	TSL6	TSL5	TSL4	TSL3	TSL2	TSL1	TSL0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TSN15	TSN14	TSN13	TSN12	TSN11	TSN10	TSN9	TSN8	TSN7	TSN6	TSN5	TSN4	TSN3	TSN2	TSN1	TSN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 828. Timestamp Status register (TSS1)

Table 568. TSS1 field descriptions

Field	Description
31:16 TSL[15:0]	Timestamp Lost Each Timestamp register (TS0-TS15) is assigned one bit. The bits are set when the timestamp stored in the related timestamp register was overwritten before it was read. Reading a Timestamp register resets the related bit.
15:0 TSN[15:0]	Timestamp New Each timestamp register (TS0-TS15) is assigned one bit. The bits are set when a timestamp was stored in the related timestamp register. Reading a timestamp register resets the related bit.

49.2.2.4 Timestamp Status 2 register (TSS2)

Offset: 0x016C Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ITBG		NTSG		0	0	0	0	0	0	0	0	TSP			
W																
Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 829. Timestamp Status register (TSS2)

Table 569. TSS2 field descriptions

Field	Description
15:14 ITBG	Internal Timebase and SOF select Generic Constant value of generic parameter internal_tb_sof_g. 00 no SOF option, no internal timebase 01 no SOF option, internal timebase (default) 10 SOF option, no internal timebase 11 SOF option, internal timebase
13:12 NTSG	Number of Timestamps Generic Constant value of generic parameter number_ts_g. 00 4 timestamp registers TS0-TS3 01 8 timestamp registers TS0-TS7 (default) 10 16 timestamp registers TS0-TS15 11 not a valid value
3:0 TSP	Timestamp Pointer The Timestamp Pointer is incremented by one each time a timestamp is captured. From its maximum value (3, 7, or 15 depending on number_ts_g), it is incremented to 0. Value also signalled on output m_can_tsp[3:0].

49.2.2.5 Timestamp n register (TSn)

Depending on the value of generic number_ts_g, the TSU stores up to 16 captured timestamps. The resolution in time depends on the configuration of TSCFG.TBPRE (internal timebase) or on the configuration of the external timebase connected to tsu_tbin[31:0].

Offset: 0x0170 +n*0x4 (n = 0 to 15) Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 830. Timestamp register (TSn)

Table 570. TSn field descriptions

Field	Description
31:0 TS	Timestamp Word TSn n = 3,7,15 depending on generic number_ts_g.

49.2.2.6 Actual Timebase (ATB)

The value of ATB is also visible at output tsu_tbout[31:0].

Offset: 0x01B0 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 831. Actual Timebase (ATB)

Table 571. ATB field descriptions

Field	Description
31:0 TB	Timebase for timestamp generation

Internal Timebase

Only available when the internal timebase has been included during TSU synthesis. To use the TSU's internal timebase, it has to be enabled via the Timebase Counter Select (TSCFG.TBCS = 0).

The internal time base consists of two counters, the 32-bit free running timebase counter and an 8-bit prescaler both implemented as up counters.

While the TSU is enabled, the prescaler increments each clock cycle. When the prescaler has reached at least TSCFG.TBPRE while the TSU is enabled (TSCFG.TSUE = 1), the prescaler is reset to zero and the timebase counter is incremented.

The TSU's internal timebase is restarted by writing to register ATB. This resets the timebase counter and the prescaler to zero. When the TSU is disabled (TSCFG.TSUE = 0), both counters are reset too.

External Timebase

There are two cases where an external timebase is used. In the first case there is no internal time base available because the internal timebase counter and prescaler have been excluded during synthesis. In this case TSCFG.TBCS is fixed to 1.

In the second case the internal timebase is disabled by software via the Timebase Counter Select (TSCFG.TBCS = 1).

In both cases the timebase value is taken from input tsu_tbin[31:0].

49.3 Functional description

49.3.1 M_CAN configuration

To enable hardware timestamping together with the TSU, M_CAN's configuration bit CCCR.UTSU has to be configured to 1.

In case CCCR.UTSU = 0 (default value), the M_CAN's internal 16-bit timestamp counter is used, if enabled.

49.3.1.1 Reception of timestamped messages

To configure the M_CAN for reception of timestamped messages, a Standard/Extended Message ID Filter Element has to be set up by configuration of S0.SSYNC = 1 respectively F1.ESYNC = 1. In case the filter element matches the received frame and the received frame is valid, M_CAN output pin m_can_tsrx is activated at EOF.

A pulse at the TSU input m_can_tsrx triggers the storage of the value of the TSU's Actual Timebase ATB to the TSU's timestamp register selected by the Timestamp Pointer TSS2.TSP[3:0]. At the same pulse, the M_CAN captures the TSP value from the TSU output m_can_tsp[3:0]. The TSP value is written to the M_CAN's R1.RXTSP[3:0] of the related Rx Buffer or Rx FIFO element.

49.3.1.2 Transmission of timestamped messages

In case bits T1.TSCE and T1.EFC of a Tx Buffer element are set, a successful transmission of that Tx Buffer generates a pulse at the TSU input m_can_tstx that triggers the storage of the value of the TSU's Actual Timebase ATB to the TSU's timestamp register selected by the Timestamp Pointer TSS2.TSP[3:0]. At the same pulse, the M_CAN captures the Timestamp Pointer value from the TSU output m_can_tsp[3:0]. The TSP value is written to the M_CAN's E1.TXTSP[3:0] of the related Tx Event FIFO element.

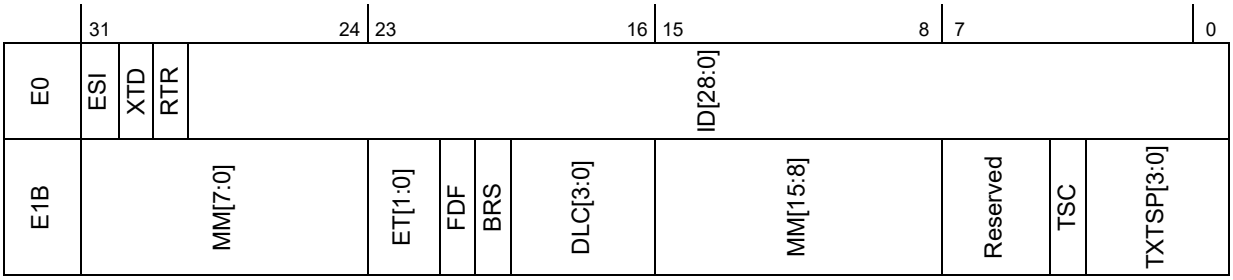


Figure 832. Tx Event FIFO Element with TSU enabled (CCCR.UTSU = 1)

49.3.2 TSU operation

49.3.2.1 TSU configuration

The TSU is configured via register TSCFG. Register TSCFG is only writable while input m_can_cce = 1.

With TSCFG.TSUE the TSU is enabled. Default after reset is disabled. The Timebase Prescaler TSCFG.TBPRE[7:0] is used to adapt the count rate of the TSU's internal counter, if enabled, to the requirements of the application.

Either the internal timebase or an external timebase input can be selected for timestamping via TSCFG.TBCS. In case TSCFG.TBCS = 0, the internal free running timebase counter is used. In case TSCFG.TBCS = 1, the timestamp value is taken from the Timebase Input tsu_tbin[31:0]. When the internal timebase has been excluded during synthesis, there is no internal timebase counter available and TSCFG.TBCS is fixed to 1.

Note: The maximum value of m_can_tsp[3:0], R1B.RXTSP[3:0], and E1B.TXTSP[3:0] depends on the number of available Timestamp registers as selected by generic parameter number_ts_g. This generic is used for synthesis and cannot be changed within an implementation.

Table 572. Configuration of generic parameter number_ts_g

number_ts_g	Timestamp registers	Maximum value
0	4	0011
1	8	0111
2	16	1111

TSCFG.SCP configures the point in time when a timestamp is captured for Sync messages (0: at EOF, 1: at SOF of the received/transmitted Sync message).

When TSCFG.SCP = 0, the timestamp is captured directly from the Actual Timebase ATB to the timestamp register TSn selected by m_can_tsp[3:0] when the message gets valid at EOF. This configuration is intended for AUTOSAR conformance applications.

When TSCFG.SCP = 1, the Actual Timebase ATB is captured into a temporary buffer at each SOF. When the message gets valid at EOF the timestamp is copied from the temporary buffer to the timestamp register TSn selected by m_can_tsp[3:0].

Note: Configuration bit TSCFG.SCP is available only when generic parameter internal_tb_sof_g was set to 2 or 3 during synthesis.

49.3.2.2 Handling of timestamps

Depending on the configuration of generic parameter number_ts_g, the TSU stores up to sixteen 32-bit timestamps (see *Table 572*). They are written to TS0 to TSn in a cyclic manner.

Each timestamp register has assigned two status bits:

- TSS1.TSNn is set whenever a timestamp was stored to the related timestamp register TSn.
- TSS1.TSLn is set when a new timestamp was stored to the related timestamp register TSn before the previously store timestamp has been read out.

Reading a Timestamp register resets the related TSS1 bits.

49.3.2.3 Capturing of timestamps

The capturing of timestamps is triggered by a pulse at the timestamp received input m_can_tsrx or the timestamp transmitted input m_can_tstx. This pulse causes the value of the Actual Timebase ATB (or, if TSCFG.SCP = 1, the temporary buffer containing the value captured at SOF) to be stored into Timestamp register TSn, with n the actual value of the Timestamp Pointer TSS2.TSP[3:0]. The timestamp pointer output m_can_tsp[3:0] signals the actual Timestamp Pointer value to the M_CAN. The Timestamp Pointer is incremented

after capturing the timestamp. For message reception, the value of m_can_tsp[3:0] is stored on position R1B.RXTSP[3:0] of the M_CAN's Rx Buffer or Rx FIFO element to which the message that triggered timestamp capturing is stored. R1B.TSC is set to 1 when a timestamp has been captured by the TSU and R1B.RXTSP[3:0] holds a valid timestamp pointer.

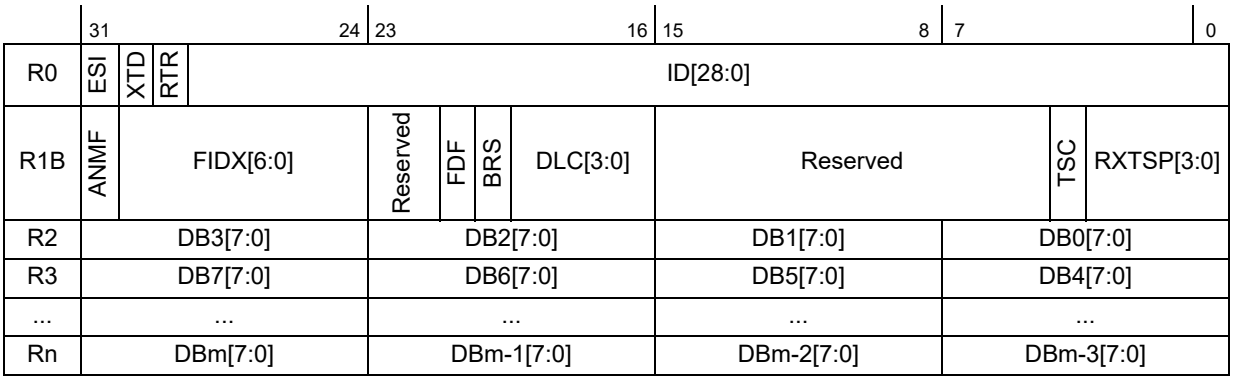


Figure 833. Rx Buffer and FIFO Element with TSU enabled (CCCR.UTSU = 1)

For message transmission, the value of m_can_tsp[3:0] is stored on position E1B.TXTSP[3:0] of the M_CAN's Tx Event FIFO element related to the Tx Buffer from which the message that triggered timestamp capturing was transmitted. E1B.TSC is set to 1 when a timestamp has been captured by the TSU and E1B.TXTSP[3:0] holds a valid timestamp pointer.

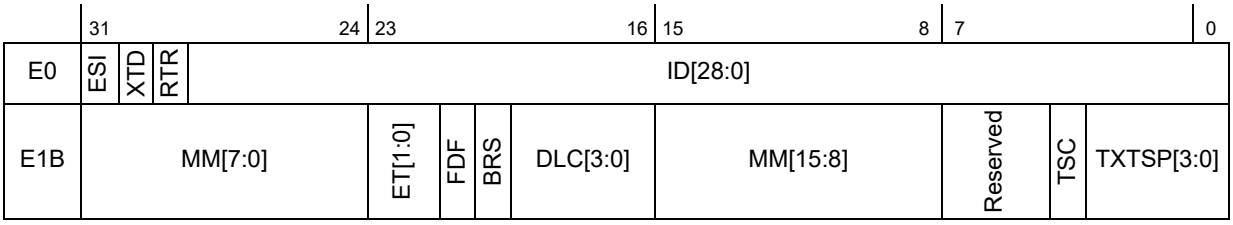
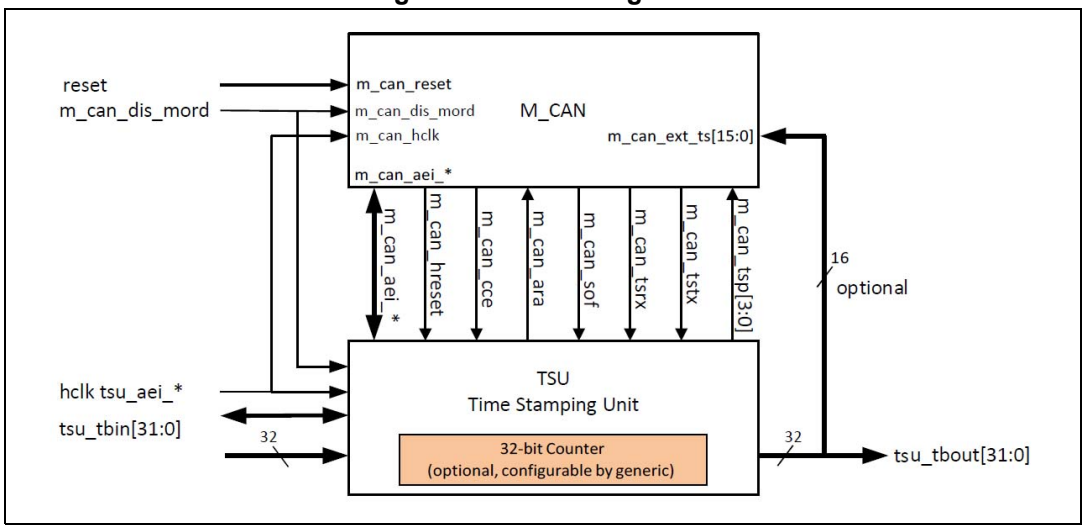


Figure 834. Tx Event FIFO Element with TSU enabled (CCCR.UTSU = 1)

49.4 TSU integration

The following figure shows how the TSU is integrated together with the M_CAN.

Figure 835. TSU integration



For configuration, control, and status monitoring, the TSU is connected to the Host CPU via tsu_aei_*. As shown in *Table 835*, the attached M_CAN's Generic Slave Interface is connected to the Host CPU through the TSU via m_can_aei_*. The M_CAN module select m_can_aei_sel is activated only in case tsu_aei_addr[9:2] is below 0x0160.

The use of Timebase Input tsu_tbin[31:0] and Timebase Output tsu_tbout[31:0] depends on the TSU's implementation and configuration.

49.4.1 TSU with internal 32-bit counter as timebase

During hardware synthesis the TSU can be configured via generic parameter to hold an internal 32-bit counter for timestamp generation. In this case the Timebase Output tsu_tbout[31:0] can be used to supply other TSUs on the device with the same timebase by connecting it to their Timebase Input(s) tsu_tbin[31:0].

Optionally, 16 bits of the Timebase Output tsu_tbout[31:0] can be connected to the External Timestamp Vector input m_can_ext_ts[15:0] of the M_CAN to be used with the M_CAN's internal timestamping mechanism.

49.4.2 TSU without internal timebase

In case the TSU was synthesized without internal 32-bit counter, the timebase for timestamping is taken from the Timebase Input tsu_tbin[31:0], connected to another TSU or to an external 32-bit timer.

49.5 New features of M_CAN

49.5.1 Generic parameter

In case a TSU is connected to an M_CAN, the M_CAN's generic parameter connected_tsu_g has to be configured to 1.

49.5.2 Additional interface signals

To support hardware timestamping by the Timestamping Unit TSU, the M_CAN has to supply the following additional signals at the module interface.

49.5.2.1 Reset synchronized to HCLK domain output (m_can_hreset)

Reset signal synchronized to m_can_hclk.

49.5.2.2 M_CAN configuration change output (m_can_cce)

Signals that configuration change for the M_CAN is enabled (CCCR.CCE = 1). Write access to register TSCFG is only possible while m_can_cce = 1.

49.5.2.3 Timestamp message received output (m_can_tsrx)

This signal is activated at EOF when the M_CAN has received a valid message for which a timestamp is needed. For reception of timestamp messages, filter element(s) have to be configured (refer to *Section 49.3.1.1: Reception of timestamped messages*).

49.5.2.4 Timestamp message transmitted output (m_can_tstx)

This signal is activated at EOF when the M_CAN has successfully transmitted a message for which a timestamp is needed. For transmission of timestamp messages the timestamp capture enable bit T1.TSCE of the respective M_CAN Tx Buffer element has to be set. In addition storage of Tx events in the Tx Event FIFO must be enabled (T1.EFC = 1).

49.5.2.5 Start of frame trigger output (m_can_sof)

To enable timestamp capturing at SOF the M_CAN has to supply a trigger output that is activated when SOF of a frame to be transmitted or received is reached.

49.5.2.6 Timestamp pointer input (m_can_tsp[3:0])

Signals the number of the TSU's Timestamp register which holds the timestamp captured by the last timestamping event to the M_CAN. The timestamp pointer operates in a cyclic manner. The maximum value depends on the configuration of generic parameter number_ts_g used for synthesis of the TSU (refer to *Table 572*).

For received timestamp messages this value is stored to R1B.RXTSP[3:0] of the respective Rx Buffer or Rx FIFO element.

For transmitted timestamp messages this value is written to E1B.TXTSP[3:0] of the respective Tx Event FIFO element.

49.5.2.7 Access to reserved address input (m_can_ara[1:0])

The M_CAN needs two additional inputs to connect the TSU and/or DMU outputs signaling an access to one of the reserved addresses of the respective add-on module.

49.5.3 Timestamp control

To enable M_CAN timestamp handling together with an external TSU module, an additional configuration bit CCCR.UTSU (Use TSU) is introduced in the CC Control register of the M_CAN when generic parameter connected_tsu_g = 1.

In case CCCR.UTSU = 0 (default value), the M_CAN's internal 16-bit timestamp counter is used, if enabled. With CCCR.UTSU = 1 the TSU is used for timestamp generation.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NISO(1)	TXP(1)	EFBI(1)	PXHD(1)	WMM(1)	UTSU(1)	BRSE(1)	FDOE(1)	TEST(2)	DAR(1)	MON(2)	CSR	CSA	ASM(2)	CCE(1)	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 836. Modified M_CAN CC Control Register (CCCR)

1. This bit is read protected write.
2. This bit is read protected set.

49.5.4 Modification of Tx buffer element

To support hardware timestamping by the TSU, bit T1.TSCE is added to the M_CAN's TX Buffer element as shown in *Figure 837*. When bit T1.TSCE (Timestamp Capture Enable) is set, a successful transmission of the Tx Buffer element triggers the capture of the TSU's timebase value as timestamp.

	31 24	23 16	15 8	7 0
T0	ESI, XTD, RTR, ID[28:0]			
T1B	MM[7:0]	EFC, TSCE, FDF, BRS, DLC[3:0]	MM[15:8]	Reserved
T2	DB3[7:0]	DB2[7:0]	DB1[7:0]	DB0[7:0]
T3	DB7[7:0]	DB6[7:0]	DB5[7:0]	DB4[7:0]
...	...	...	...	...
Tn	DBm[7:0]	DBm-1[7:0]	DBm-2[7:0]	DBm-3[7:0]

Figure 837. Modified Tx Buffer Element

49.5.5 Modification of Rx Buffer and FIFO element

In case the M_CAN is used together with the TSU, the Rx Buffer and FIFO Element is modified according to *Figure 833*.

49.5.6 Modification of Tx event FIFO element

In case the M_CAN is used together with the TSU, the Tx Event FIFO Element is modified according to *Figure 834*.

49.5.7 Modification of standard message ID Filter Element

To enable filtering for Sync messages bit SSYNC is added at bit position 15 of S0.

49.5.8 Modification of extended message ID Filter Element

To enable filtering for Sync messages bit ESYNC is added at bit position 29 of F1.

50 Hardware semaphore (HSEM2)

50.1 Hardware semaphore introduction

The hardware semaphore block provides 32 (32-bit) register based semaphores.

The semaphores can be used to ensure synchronization between different processes running between different cores. The HSEM2 provides a non blocking mechanism to lock semaphores in an atomic way. The following functions are provided:

- Locking a semaphore can be done in two ways:
  - 2-step lock: by writing COREID and PROCID to the semaphore, followed by a read check
  - 1-step lock: by reading the COREID from the semaphore
- Interrupt generation when a semaphore is freed
  - Each semaphore may generate an interrupt on one of the interrupt lines
- Semaphore clear protection
  - A semaphore is only cleared when COREID and PROCID match
- Global semaphore clear per COREID

50.2 Hardware semaphore main features

The HSEM2 includes the following features:

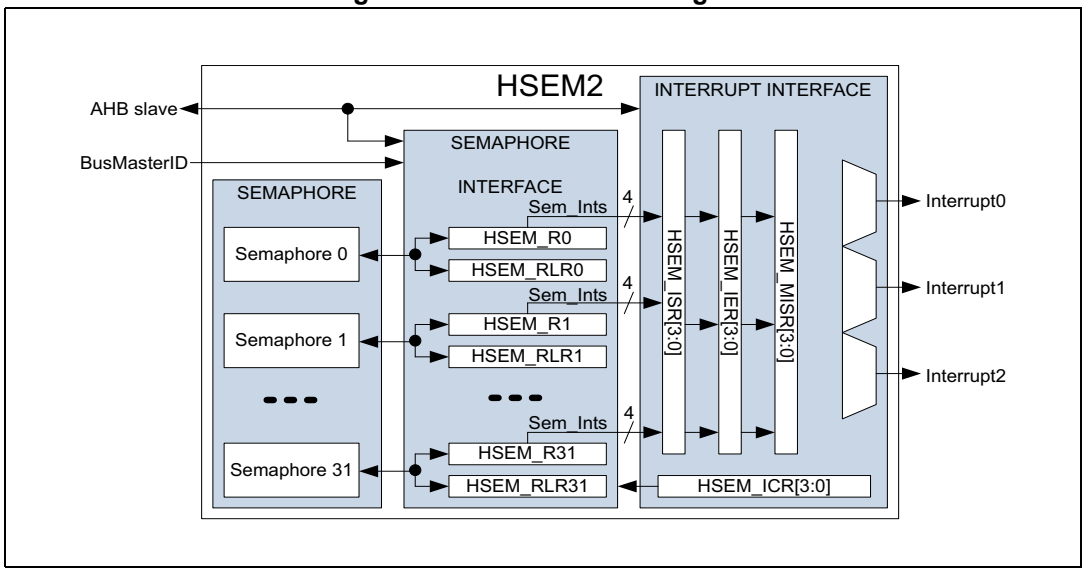
- 32 (32-bit) semaphores
- 8-bit PROCID
- 4-bit COREID
- 3 interrupt lines
- Lock indication

50.3 HSEM2 block diagram

As shown in *Figure 838*, the HSEM2 is based on three sub-blocks:

- The semaphore block containing the semaphore status and IDs
- The semaphore interface block providing AHB access to the semaphore via the HSEM2_Rx and HSEM2_RLRx registers
- The interrupt interface block providing control for the interrupts via the HSEM2_ISRn, HSEM2_IERn, HSEM2_MISRn, and HSEM2_ICRn registers

Figure 838. HSEM2 block diagram



50.4 HSEM2 functional description

50.4.1 HSEM2 lock procedures

There are two lock procedures:

- 2-step (write) lock
- 1-step (read) lock

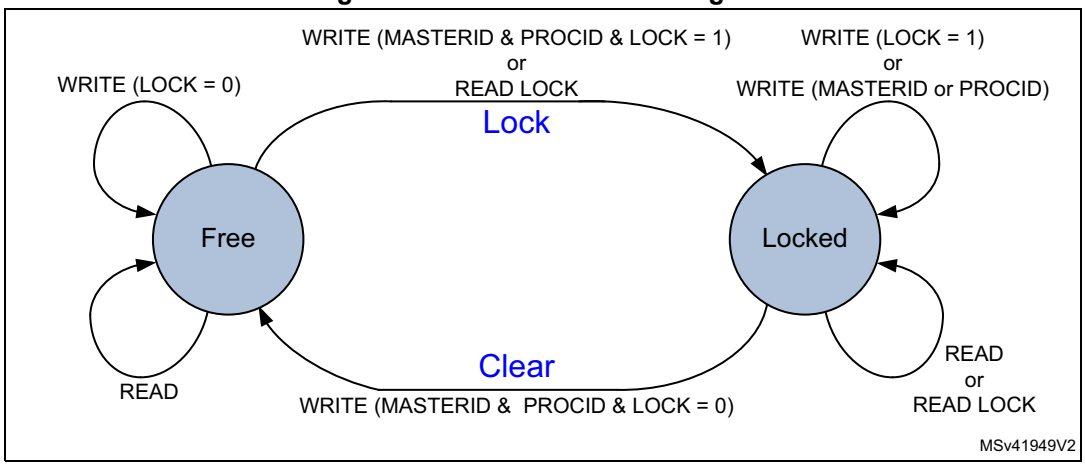
The semaphore is free when its LOCK bit is 0. In this case, the COREID and PROCID are also 0. When the LOCK bit is 1, the semaphore is locked and the COREID indicates which AHB bus master locked it. The PROCID indicates which process of that AHB bus master locked the semaphore.

When write locking a semaphore, the COREID is taken from the master ID and the PROCID is taken from the write data. When read locking the semaphore, the COREID is taken from the AHB bus master ID, and the PROCID is zero. There is no PROCID available with the 1-step (read) lock.

The COREID is taken from the AHB bus master ID. The PROCID is written by the software of that AHB bus master. Each AHB bus master process must have a unique PROCID. PROCID is only available in the 2-step lock procedure.

The two procedures (1-step and 2-step) can be used concurrently.

Figure 839. Procedure state diagram



2-step (write) lock procedure

The 2-step lock procedure consists in a write to lock the semaphore, followed by a read to check if the lock has been successful, carried out from the HSEM2_Rx register

- Write semaphore with PROCID and COREID, and LOCK = 1

  Lock is put in place when semaphore is free at write time.
- Read-back the semaphore

  The software checks the lock status, if PROCID and COREID match, then the lock is confirmed.
- Else retry (the semaphore has been locked by another AHB bus master or process)

A semaphore can only be locked when it is free.

A semaphore can be locked when the PROCID = 0.

Consecutive write attempts with LOCK = 1 to a locked semaphore, are ignored.

1-step (read) lock procedure

The 1-step procedure consists in a read to lock and check the semaphore in a single step from the HSEM2_RLRx register.

- Read Lock semaphore with COREID.
- If read COREID matches and PROCID = 0, then the lock is put in place. If COREID matches and PROCID is not 0, this means that another process from the same COREID locked the semaphore with a 2-step (write) procedure.
- Else retry (the semaphore has been locked by another AHB bus master or process)

A semaphore can only be locked when it is free. When read locking a free semaphore, PROCID is 0. Read locking a locked semaphore returns the COREID and PROCID that locked it. All read locks, including the first one that locks the semaphore, return the COREID that locks or locked the semaphore.

If multiple processes of the same AHB bus master use the 1-step procedure, all processes using the same semaphore read the same status. When only one process locks the semaphore, each process of that AHB bus master reads the semaphore as locked by itself with the COREID.

50.4.2 HSEM2 write/read/readlock register address

For each semaphore, two AHB register addresses are provided, separated in two banks of 0x100.

In the first register address bank the semaphore can be written (locked/cleared) and read through the HSEM2_Rx registers.

In the second register address bank the semaphore can be read (locked) through the HSEM2_RLRx registers.

50.4.3 HSEM2 clear procedures

Clearing a semaphore is a protected process, to prevent accidental clearing by an AHB bus master or by a process that does not have the semaphore lock right. The semaphore clear procedure consists in writing to the semaphore with the corresponding COREID and PROCID and LOCK = 0. When cleared, the semaphore LOCK, the COREID, and the PROCID are all 0.

When cleared, an interrupt may be generated to signal the event. To this end, the semaphore interrupt must be enabled.

The clear procedure consists in a write to the semaphore HSEM2_Rx registers:

- Write semaphore with PROCID and COREID, LOCK = 0.
- If PROCID and COREID match, semaphore is freed, and an interrupt may be generated when enabled.
- Else write is ignored, semaphore remains locked and no interrupts are generated (the semaphore is locked by another AHB bus master or process).

If multiple processes of the same AHB bus master use the 1-step lock procedure (PROCID = 0), all processes using the same semaphore clear the semaphore also for the other processes of that AHB bus master.

50.4.4 HSEM2 COREID semaphore clear

All semaphores locked by an AHB bus master can be cleared all at once by using the HSEM2_CR register.

The procedure to clear all semaphores locked by an AHB bus master is the following:

- Write COREID and correct KEY value. All locked semaphore with a matching COREID are cleared (set to free), and may generate an interrupt when enabled.

This procedure may be used in case of an incorrect functioning AHB bus master, where another AHB bus master can free the locked semaphores by writing the incorrect functioning COREID into the HSEM2_CR register with the correct KEY value. This clears all locked semaphores with a matching COREID.

An interrupt may be generated for the semaphore(s) that become free. To this end, the semaphore interrupt must be enabled in the HSEM2_IERn registers.

50.4.5 HSEM2 interrupts

There are 3 interrupt lines allowing each of the semaphores to generate an interrupt.

Each of these interrupt lines provides the following features:

- Interrupt enable per semaphore
- Interrupt clear per semaphore
- Interrupt status per semaphore
- Masked interrupt status per semaphore

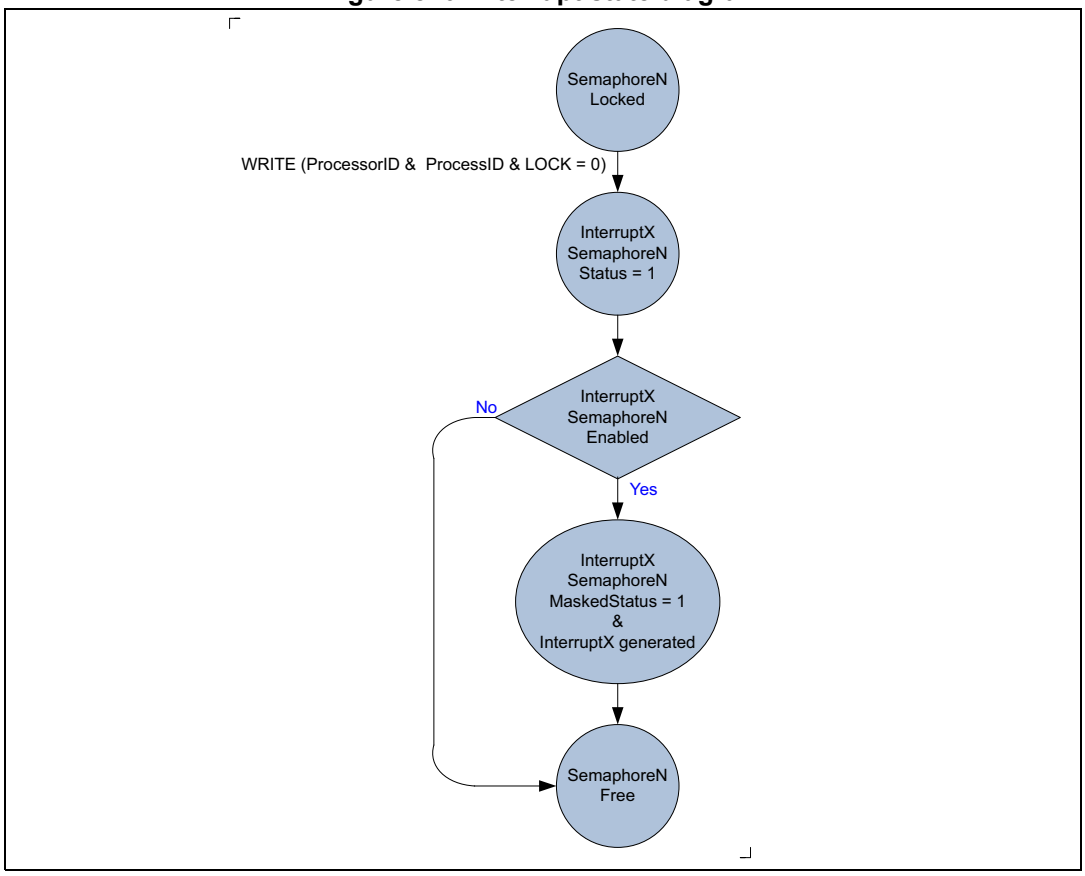
With the interrupt enable (HSEM2_IERn), the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts do not set the masked interrupt status for that semaphore, and do not generate an interrupt on the interrupt line.

The interrupt clear (HSEM2_ICRn) clears the interrupt status and masked interrupt status of the associated semaphore for the interrupt line.

The interrupt status (HSEM2_ISRn) mirrors the semaphore interrupt status of the interrupt line before the enable.

The masked interrupt status (HSEM2_MISRn) only mirrors the semaphore interrupt status of the enabled semaphore interrupts on the interrupt line. All masked interrupt status of the enabled semaphore must be cleared in order to clear the interrupt line.

Figure 840. Interrupt state diagram



The procedure to get an interrupt when a semaphore becomes free is described hereafter.

Try to lock the semaphore x

- If the semaphore lock is obtained, no interrupt is needed.
- If the semaphore lock fails:

  Clear pending semaphore x interrupt status for the interrupt line hsem2_intn_it in HSEM2_ICRn.

  Re-try to lock the semaphore x again:
  - If the semaphore lock is obtained, no interrupt is needed (semaphore has been freed between first try to lock it and clear semaphore interrupt status).
  - If the semaphore lock fails, enable the semaphore x interrupt for the interrupt line hsem2_intn_it in HSEM2_IERn.

On the semaphore x free interrupt, try to lock the semaphore x

- If the semaphore lock is obtained:

  Disable the semaphore x interrupt for the interrupt line hsem2_intn_it in HSEM2_IERn.

  Clear pending semaphore x interrupt status for the interrupt line hsem2_intn_it in HSEM2_ICRn.
- If the semaphore x lock fails:

  Clear pending semaphore x Interrupt status for the interrupt line hsem2_intn_it in HSEM2_ICRn.

  Try again to lock the semaphore x:
  - If the semaphore lock is obtained (semaphore has been freed between first try to lock and semaphore Interrupt status clear), Disable the semaphore interrupt for the interrupt line hsem2_intn_it in HSEM2_IERn.
  - If the semaphore lock failed, wait for semaphore free interrupt.

Note: An interrupt does not lock the semaphore. After an interrupt, either the AHB bus master or the process must still perform the lock procedure to lock the semaphore.

It is possible to have multiple AHB bus masters informed by the semaphore free interrupts. Each AHB bus master gets its interrupt, and the first one to react locks the semaphore.

50.4.6 AHB bus master ID verification

The HSEM2 allows only authorized AHB bus master IDs to lock and unlock semaphores.

- The AHB bus master 2-step lock write access to the semaphore HSEM2_Rx register is checked against the valid bus master IDs.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not lock the semaphore.
- The AHB bus master 1-step lock read access from the semaphore HSEM2_RLRx register is checked against the valid bus master IDs.
  - An unauthorized AHB bus master ID read from HSEM2_RLRx returns the following Read data depending on the semaphore status:
    - when the semaphore is free, it returns all 0.
    - when the semaphore has been locked before, it returns the HSEM2_RLRx data.

- The COREID semaphore clear write access to the HSEM2_CR register is checked against the valid bus master IDs. Only the valid bus master IDs can write to the HSEM2_CR register and clear any of the COREID semaphores.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not clear the COREID semaphore.

Table 573 details the relation between bus master/CPU and COREID.

Table 573. Authorized AHB bus master IDs

Parameter	Description	HSEM2_CR, COREID
MASTERID1	Core_1 master	1000
MASTERID2	Core_2 master	0001
MASTERID3	HSM master	0010

Note: Accesses from unauthorized AHB bus master IDs to other registers are granted.

50.5 HSEM2 registers

Registers must be accessed using word format. Byte and half-word accesses are ignored and have no effect on the semaphores. Byte and half-word read accesses always return 0. Byte and half-word accesses do not generate a bus error.

50.5.1 HSEM2 memory map

Table 574. HSEM2 register memory map

Offset	Register name
0x000 + 0x4 * x (x = 0 to 31)	*HSEM2 register semaphore x (HSEM2_Rx)*
0x080 + 0x004 * x (x = 0 to 31)	*HSEM2 read lock register semaphore x (HSEM2_RLRx)*
0x100 + n*0x10 (n=0 to 2)	*HSEM2 interrupt enable register (HSEM2_IERn) (n=0 to 2)*
0x104 + n*0x10 (n=0 to 2)	*HSEM2 interrupt clear register (HSEM2_ICRn) (n=0 to 2)*
0x108 + n*0x10 (n=0 to 2)	*HSEM2 interrupt status register (HSEM2_ISRn) (n=0 to 2)*
0x10C + n*0x10 (n=0 to 2)	*HSEM2 interrupt status register (HSEM2_MISRn) (n=0 to 2)*
0x140	*HSEM2 clear register (HSEM2_CR)*
0x144	*HSEM2 interrupt clear register (HSEM2_KEYR)*

50.5.2 HSEM2 register semaphore x (HSEM2_Rx)

Address offset: 0x000 + 0x4 * x (x = 0 to 31)

Reset value: 0x0000 0000

The HSEM2_Rx must be used to perform a 2-step write lock and read back. Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	COREID[3:0]				PROCID[7:0]							
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock indication

This bit can be written and read by software.

0: On write free semaphore (only when COREID and PROCID match), on read semaphore is free.

1: On write try to lock semaphore, on read semaphore is locked.

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]:** Semaphore COREID

Written by software, when the semaphore is free and the LOCK bit is at the same time written to 1, the COREID is written only when the bus ID of the AHB bus master writing the semaphore matches.

When the semaphore is cleared (LOCK bit written to 0 and AHB bus master ID matched COREID), the COREID is cleared to 0.

When the semaphore is cleared (LOCK bit written to 0 and AHB bus master ID does not match COREID), the COREID is not be affected.

Write when LOCK bit is already 1 (semaphore locked), the COREID is not affected.

A read returns the stored COREID value.

Bits 7:0 **PROCID[7:0]:** Semaphore PROCID

Written by software, when the semaphore is free and the lock bit is written to 1, the PROCID is set to the written data.

When the semaphore is cleared, (LOCK bit written to 0), the PROCID is cleared to 0.

Write when LOCK bit is already 1 (semaphore locked), the PROCID is not affected.

A read returns the programmed PROCID value.

50.5.3 HSEM2 read lock register semaphore x (HSEM2_RLRx)

Address offset: 0x080 + 0x004 * x (x = 0 to 31)

Reset value: 0x0000 0000

Accesses the same physical bits as HSEM2_Rx. The HSEM2_RLRx must be used to perform a 1-step read lock. Only read accesses with authorized AHB bus master IDs are granted. Read accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	COREID[3:0]				PROCID[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **LOCK:** Lock indication

This bit is read only by software at this address. A read with a valid bus master ID always returns 1.

When the semaphore is free and the software performs a read, the hardware sets the semaphore to locked.

When the semaphore is locked and the software performs a read, the LOCK bit is not effected.

0: Semaphore free

1: Semaphore locked

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]:** Semaphore COREID

This field is read only by software at this address.

On a read, when the semaphore is free, the hardware sets the COREID to the AHB bus master ID reading the semaphore. The COREID of the AHB bus master locking the semaphore, is read.

On a read when the semaphore is locked, this field returns the COREID of the AHB bus master that has locked the semaphore.

Bits 7:0 **PROCID[7:0]:** Semaphore processor ID

This field is read only by software at this address.

On a read when the semaphore is free, this filed returns 0.

On a read when the semaphore is locked, this field returns the processor ID of the process that locked the semaphore.

50.5.4 HSEM2 interrupt enable register (HSEM2_IERn) (n=0 to 2)

Address offset: 0x100 + n*0x10 (n=0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ISE[31:0]:** Interrupt semaphore x enable bit
This bit is read and written by software.
0: Interrupt generation for semaphore x disabled (masked)
1: Interrupt generation for semaphore x enabled (not masked)

50.5.5 HSEM2 interrupt clear register (HSEM2_ICRn) (n=0 to 2)

Address offset: 0x104 + n*0x10 (n=0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISC[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISC[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ISC[31:0]:** Interrupt semaphore x clear bit
This bit is read and written by software, and is always read 0.
0: Interrupt semaphore x status and masked status not affected
1: Interrupt semaphore x status and masked status cleared

50.5.6 HSEM2 interrupt status register (HSEM2_ISRn) (n=0 to 2)

Address offset: 0x108 + n*0x10 (n=0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ISF[31:0]:** Interrupt semaphore x status bit before enable (mask)

This bit is set by hardware, and reset only by software. This bit is cleared by software writing the corresponding HSEM2_ICRn bit.

0: Interrupt semaphore x status, no interrupt pending

1: Interrupt semaphore x status, interrupt pending

50.5.7 HSEM2 interrupt status register (HSEM2_MISRn) (n=0 to 2)

Address offset: 0x10C + n*0x10 (n=0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MISF[31:0]:** Masked interrupt semaphore x status bit after enable (mask)

This bit is set by hardware and read only by software. This bit is cleared by software writing the corresponding HSEM2_ICRn bit.

0: Interrupt semaphore x status after masking not pending

1: Interrupt semaphore x status after masking pending

50.5.8 HSEM2 clear register (HSEM2_CR)

Address offset: 0x140

Reset value: 0x0000 0000

Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	COREID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				w	w	w	w								

Bits 31:16 **KEY[15:0]:** Semaphore clear key
This field can be written by software and is always read 0.
If this key value does not match HSEM2_KEYR.KEY, semaphores are not effected.
If this key value matches HSEM2_KEYR.KEY, all semaphores matching the COREID are cleared to the free state.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]:** COREID of semaphores to be cleared
This field can be written by software and is always read 0.
This field indicates the COREID for which the semaphores are cleared when writing the HSEM2_CR.

Bits 7:0 Reserved, must be kept at reset value.

50.5.9 HSEM2 interrupt clear register (HSEM2_KEYR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **KEY[15:0]:** Semaphore clear key
This field can be written and read by software.
Key value to match when clearing semaphores.

Bits 15:0 Reserved, must be kept at reset value.

51 Fault collection and control unit (FCCU)

51.1 Introduction

The Fault Collection and Control Unit (FCCU) offers a systematic approach to fault detection and control. The FCCU provides a hardware channel to collect errors and to place the device into a safety state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

51.2 Features

The FCCU offers a systematic approach to fault detection and control. The distinctive features of the module are:

- 1 to 128 recoverable faults management
- HW or SW fault recovery management
- Fault detection collection
- Fault injection (fake faults; refer to the implementation-specific details)
- Bi-stable, Dual Rail and Time Switching output protocols on EOUT
- Watchdog timer for the reconfiguration phase
- Redundant collection of hardware checker (for example RCCU) results
- Redundant collection of error information from safety relevant modules on the device
- Collection of test results
- Configurable and graded fault control
- Internal (SoC) reactions
  - ALARM state: interrupt request
  - FAULT state: long functional reset request pulse, short functional reset request pulse, NMIs
- Internal reactions (independently configurable for each RF)
  - No reset reaction
  - IRQ
  - Short functional reset
  - Long functional reset
  - NMIs
- External reaction (failure is reported to the outside world via one or more output pins)
- External reaction (fault state): EOUT signaling. Failure indication via the pin(s) is controlled by the FCCU
- Configuration lock: The FCCU’s configuration is lockable
  - Permanently locked until next reset or transiently locked until a specific key is written. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (a value different from 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register

- One of the failure indication pins is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the failure indication protocol is configured to be a toggling protocol that is dual rail and time switching
- After power-on the error out pins have high impedance[c]. FCCU goes to operational state only on software request
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in bi-stable mode only). T_min = 250 µs + delta_T, with delta_T parameter being configurable by SW up to 10 ms
- The overall error detection, processing and indication time is less than 10 ms
- The actual maximum time is five safe (IRCOSC) clock cycles

Two classes of fault are identified based on the criticality and the related reactions. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable based on the fault criticality. The default configuration can be modified only in a specific FCCU state for application/test/debugging purposes. FCCU is designed for assuming that the system clock is faster than the IRC clock.

51.2.1 Standard feature

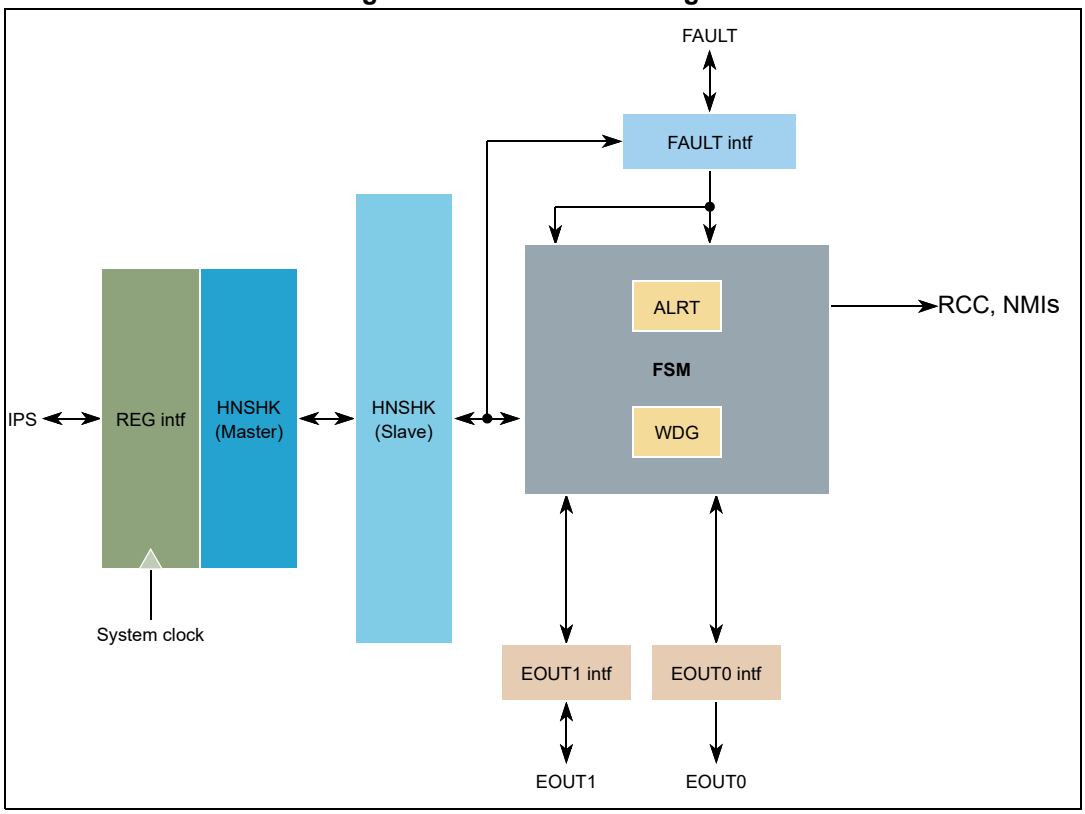
- IPS (slave bus signals) bus interface

51.3 Block diagram

The top-level diagram of the FCCU module is given in *Figure 841*.

c. Actual value depends on the SoC settings at pad level.

Figure 841. FCCU block diagram



The FCCU module includes the submodules listed in *Table 575*.

Table 575. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface.
HNSHK blocks (master and slave blocks)	Includes the FSMs to support the handshake between the REG if and the FSM unit due to the usage of two asynchronous clocks (IPS system clock and RC oscillator clock).
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: – Watchdog timer (WDG) – Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management.
EOUTx units	Implement the output stage to manage the EOUT interfaces.

51.4 Signal description

51.4.1 Pinout

The FCCU generates two external signals, EOUT[0] and EOUT[1]. These are described in *Section 51.5.9: EOUT interface*.

For the availability of these signals on a chip, refer to the device configuration chapter.

51.4.2 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers.
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers.
- Byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0] data write/read operations to any registers.
- Any other operation (free byte enables or other operations) must be avoided.

The FCCU module generates a transfer error in the following cases:

- Any write/read access executed outside the register address space of the peripheral. Note that if some registers are not implemented (for example FCCU_CFS_CFGx registers), they generate no bus transfer error when accessed in writing.
- Any write/read operation different from byte/hword/word (free byte enables or other operations) on each register.
- Any write access executed on configuration registers, when the fccu is not in the CONFIG state. Configuration registers are those registers which can be written only in CONFIG state. Refer to *Table 584: FCCU memory map*.
- Any write executed on FCCU_CFG_TO register, when the fccu is in the CONFIG state.
- Any write access to the Transient and permanent Lock Register in any mode other than supervisor access mode.

The registers of the FCCU module are accessible (read/write) in each access mode: user, supervisor or test.

51.5 Functional description

51.5.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is asserted. That is, if 0 on the fault signal indicates fault, then the status flags are valid until the fault line is ‘0’. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched in an external module at the FCCU. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the error is recoverable by itself when the error condition gets corrected.

The following types of reset are applicable:

- Destructive reset: any type of reset related to a power failure condition that implies a complete system re-initialization.
- Long functional reset: implies the Flash and digital circuitry (most of it except FCCU, STCU) initialization.
- Short functional reset: implies the digital circuitry (most of it except FCCU, STCU) initialization.

51.5.2 FSM description

The FCCU module functionality is depicted by the FSM diagram given in *Figure 842*.

Basically four states are identified with the following meaning:

- CONFIG: The configuration state is used only to modify the default configuration of the FCCU. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, recoverable fault masking) can be accessed in write mode only in the CONFIG state.
  The CONFIG state is accessible only in NORMAL state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset of the FCCU, the transient lock register is unlocked by writing 0xBC into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (a value different from 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register. After reset lifts, the state must be transiently locked (permanent lock → is unlocked and transient lock → is locked).
  The CONFIG to NORMAL state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to NORMAL appear at the same time, watchdog timeout has the priority and hence the Configuration registers (those that are writable only in CONFIG mode) are reset to their default values. Also, FCCU_CFG_TO and FCCU_EINOUT are reset to their default values, although they do not meet the condition of being a configuration register. The movement to NORMAL is made.
  The incoming faults, occurring during the configuration phase (CONFIG state) are anyway latched in order to process them when the FCCU is moved into the NORMAL state, according to the new configuration.
- NORMAL: This is the FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Succeeding one of the following events:
  - unmasked recoverable faults with the timeout disabled → the FCCU moves to the FAULT state.
  - unmasked recoverable faults with the timeout enabled → the FCCU moves to the ALARM state.
  - masked recoverable faults → the FCCU stays in NORMAL state.
- ALARM: the FCCU moves into the ALARM state when an unmasked recoverable fault occurs and the timeout is enabled. The transition to the ALARM state goes along with an interrupt request (ALARM state), if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to the FAULT state. The timeout is re-initialized if the FCCU state moves to the NORMAL state. The timeout restarts following the recovery from the FAULT state.

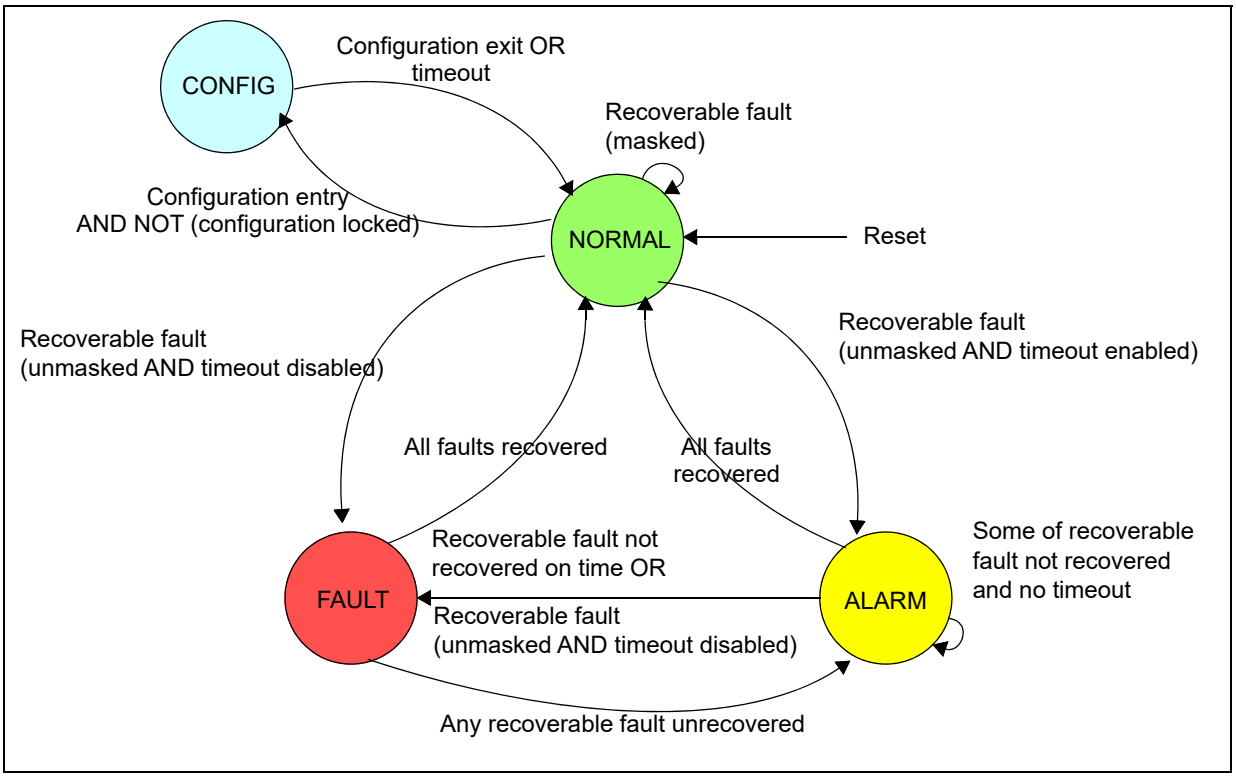
- FAULT: the FCCU moves into the FAULT state when one of the following conditions occurs:
  - timeout related to a recoverable fault when the FCCU is in the ALARM state.
  - unmasked recoverable faults with the timeout disabled.

The transition from NORMAL/ALARM state goes along with the generation of:

- NMIs interrupt (optional).
- EOUT signaling (optional).
- SW option: Soft reaction (Short functional reset request pulse if configured).
- SW option: Hard reaction (Long functional reset request pulse if configured).

Non Maskable Interrupts (NMIs) are routed only to the Safety Core.

Figure 842. FCCU state diagram



51.5.3 Reset interface

The FCCU has two input resets, as described in *Table 576* and two output resets related to the FAULT state. In FAULT state, each fault can be programmed to generate a soft or hard reaction request pulse.

In *Table 577* for each type of fault reaction (hard, soft or self-checking), the respective output reset (Long functional reset, Short functional reset) is activated. In *Table 578* for each type of reset source (external reset, POR, STCU, FCCU failure, short functional, long functional), the respective FCCU input reset state is described.

The RCC unit generates the corresponding input reset as described in *Table 576*.

Table 576. Input reset

Input reset	Description
System reset	Global reset of the FCCU synchronous to the system clock
System reset (RC oscillator)	Global reset of the FCCU synchronous to the IRCOSC

Table 577 and *Table 578* describe a typical reset strategy at system level.

Table 577. Reset reactions

FAULT reaction	Active reset	System reset signal
Hard	Long functional reset	Disabled
Soft	Short functional reset	Disabled

Table 578 shows the Reset sources.

Table 578. Reset sources

Reset source	System reset signal
External reset	Enabled
POR	Enabled
STCU	Enabled
FCCU failure	Application dependent
Functional reset	Disabled

Table 579 summarizes the RCC actions corresponding to each reset generated by FCCU.

Table 579. RCC actions

Active reset	RCC action
Long functional reset	Reset Flash and digital modules excluded FCCU, STCU
Short functional reset	Reset digital modules excluded FCCU, STCU

In reset state the FCCU is not able to collect any RF. The FCCU reset phase (POR, external reset, STCU, destructive reset, FCCU failure) must guarantee the assertion of the FCCU input resets at least for two IRCOSC clock periods.

51.5.4 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (possible for RFs only) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a RF (unmasked and with timeout disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the RF (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery) otherwise the FCCU remains in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the RF (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery) otherwise the FCCU returns in the ALARM state (if any RF is still pending and the timeout is not elapsed).

In general, no fault nesting is supported except for the RF that causes an ALARM to FAULT state transition. In this case the alarm timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer must not reload and must not start again.

51.5.5 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a RF management (ALARM state), refer to *Figure 843* and *Figure 844*, is as follows:

- RF assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - Alarm interrupt request (if configured)
  - Timeout running
- System state: RUN
- Alarm interrupt management: FAULT recovery (by SW): FCCU state transition ALARM → NORMAL

Figure 843. RF (ALARM state) recovery (a)

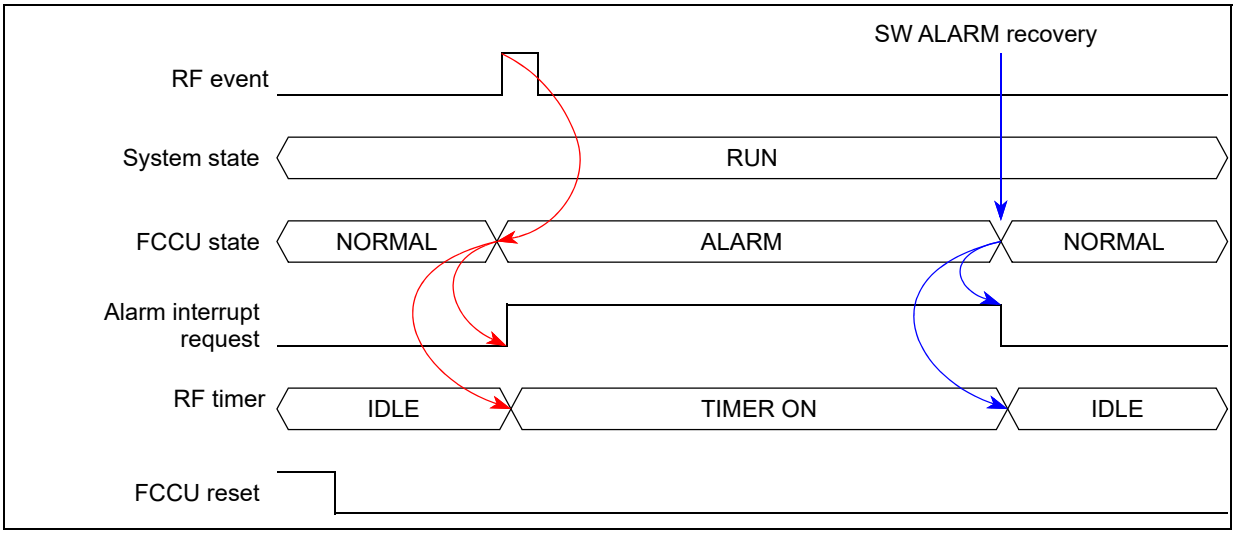
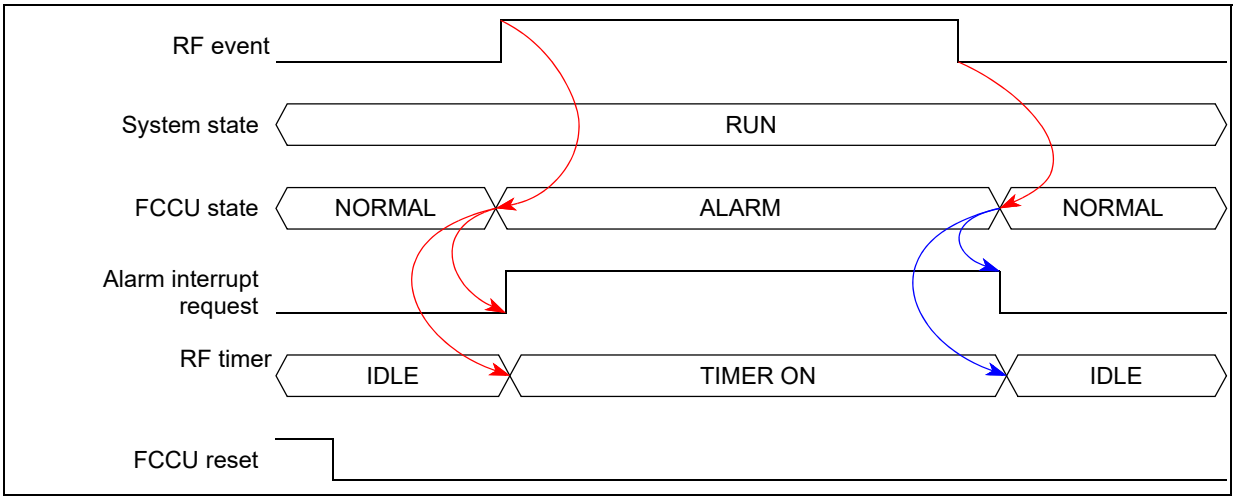


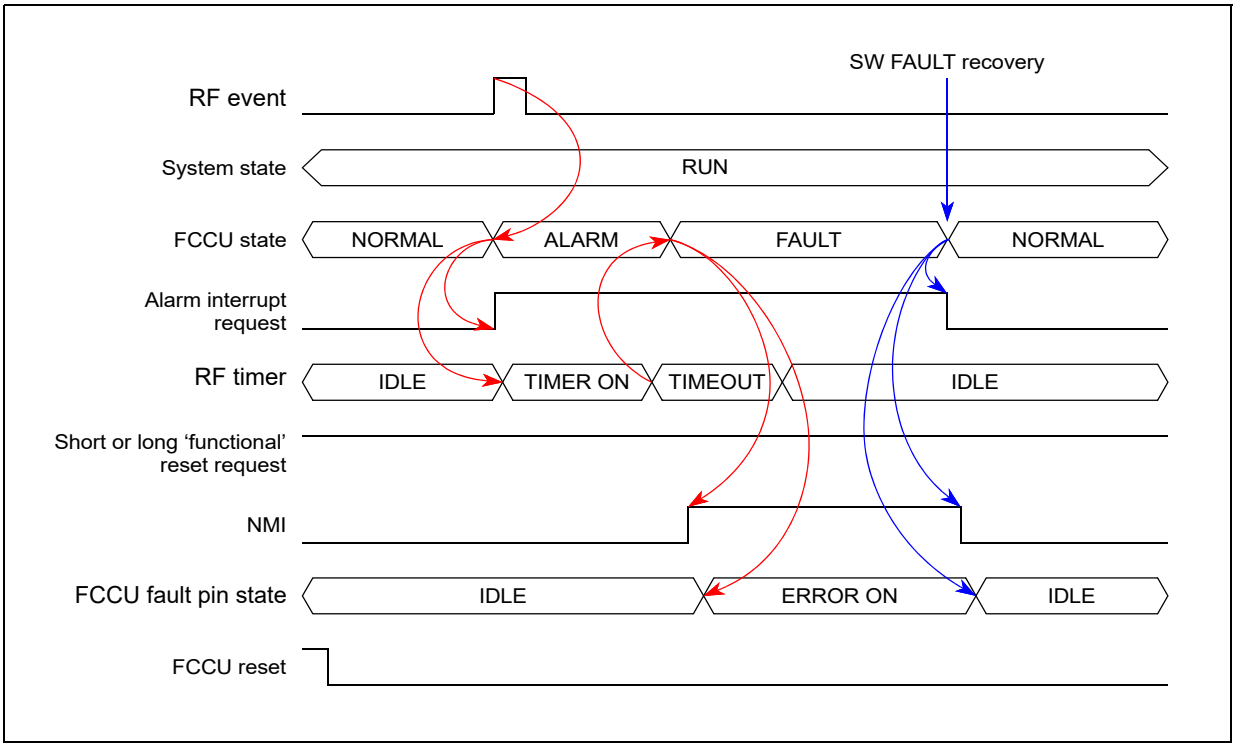
Figure 844. RF (ALARM state) recovery (b)



A typical sequence related to a RF management (ALARM → FAULT state), refer to *Figure 845*, is as follows:

- RF assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - Alarm interrupt request (if enabled)
  - Timeout running
- FCCU state transition (following the timeout trigger): ALARM → FAULT: NMIs assertion (if enabled)
- NMIs interrupt management (if enabled)
  - FAULT recovery (by SW): FCCU state transition FAULT → NORMAL

Figure 845. RF(ALARM -> FAULT state) recovery



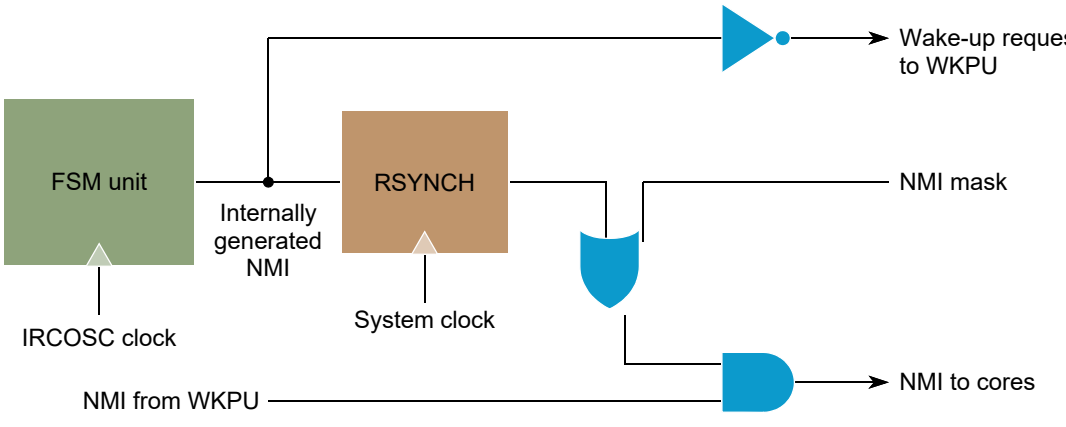
51.5.6 NMI/WKPU interface

The wake-up interface includes Non-maskable interrupt: NMI interrupt source generated by the WKPU. It is synchronous with the system clock and active low. A logical AND interconnects the NMI sources (internally generated by FCCU and by WKPU) in order to provide a single NMI output.

The NMI signal internally generated by FCCU is masked when:

- FCCU asynchronous reset
- the device is in reset state

Figure 846. NMI/WKUP scheme



Note: All NMI signals are shown active-low.

51.5.7 FAULT interface

The basic functionality of the fault interface is to provide a simple, programmable edge-sensitive interface (active high or low) to signal a fault condition to the FCCU module.

To support a fault injection mechanism an additional and optional signal is available. It is an active high signal and is asserted for a single clock cycle synchronously at the system clock. The behavior is given in *Figure 847*. The fault injection is executed by a write operation in the FCCU_RFF register.The reaction following a fake RF cannot be masked. The fault injection mechanism is optional; refer to the implementation-specific details. Refer to “FCCU failure inputs” table in Functional Safety chapter, for more details.

To support a fault clearing mechanism, in order to clear a fault latched directly in the FAULT root, an additional and optional signal (FCCU recoverable faults clear) is available. This active-high signal is asserted for a single clock cycle synchronously at the system clock. The behavior is given in *Figure 848*. The fault clearing is executed when the SW “clears” the status (flag) bits of the FCCU_RF_S*n*. The fault clearing mechanism is optional and its support must be specified at system level.

Figure 847. FAULT injection

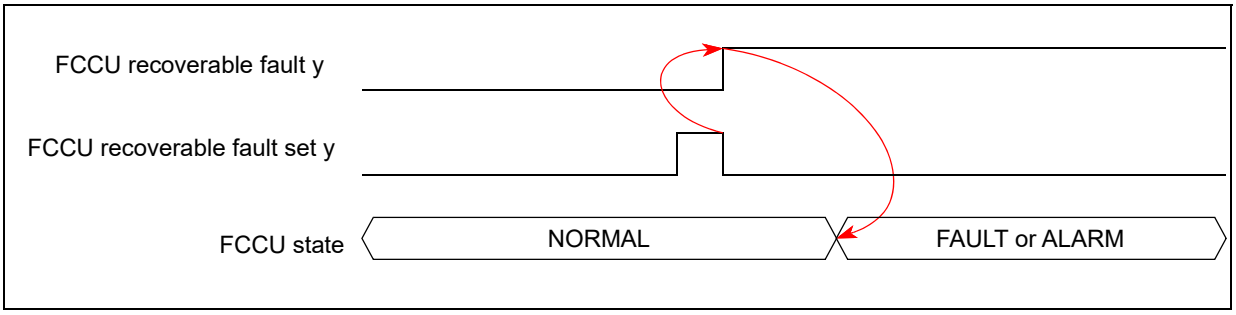
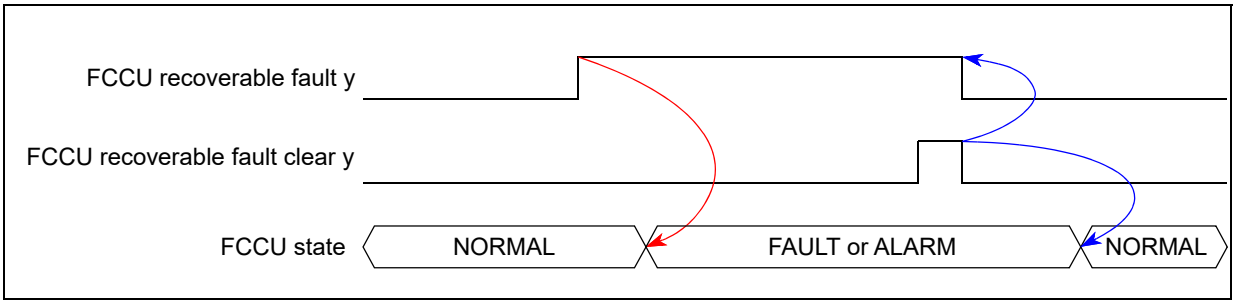


Figure 848. FAULT clearing



51.5.8 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the following FCCU fault:
  - Recoverable
- The STCU fault signals are processed by the FCCU when the SOC is rebooted following the self-testing procedure. The STCU includes also a status register that stores the self-testing results (flags).
- A mask (EOUT mask[1:0]) that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self testing procedure, depending on the STCU results, three cases are applicable:

a) STCU completes the self testing procedure successfully. The SOC reboots and the FCCU is responsible for providing a reaction.

b) STCU completes the self testing procedure with low severity failures. The FCCU is responsible for providing the proper reactions according to the fault occurred.

c) STCU completes the self testing procedure with serious failures. The FCCU or other critical parts of the SoC could not be able to provide the proper reaction. STCU must keep permanently the SoC in RESET state. This feature is optionally programmable inside the STCU.

Figure 849. STCU-FCCU (case a)

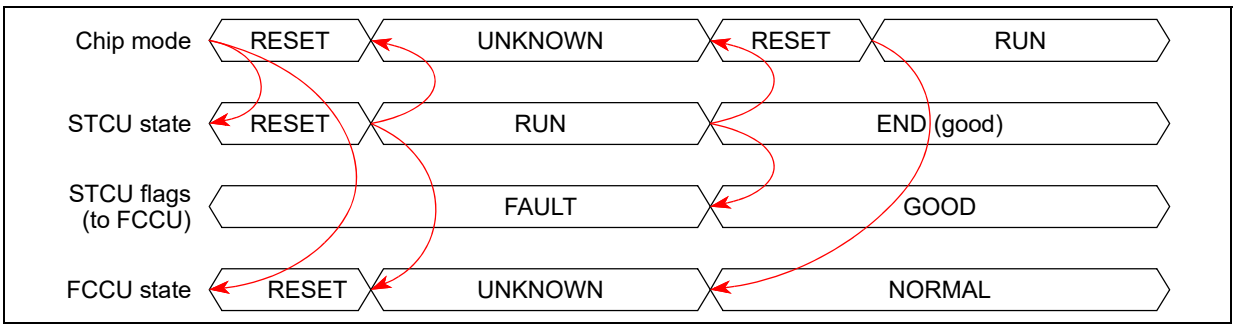


Figure 850. STCU-FCCU (case b)

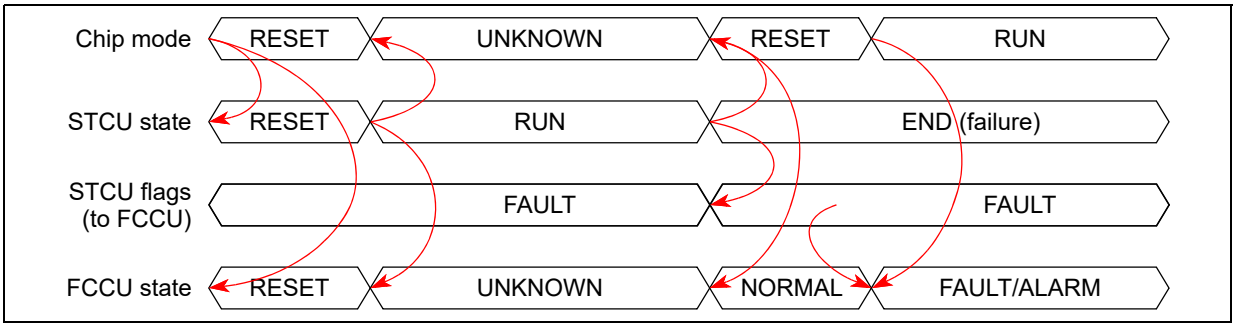
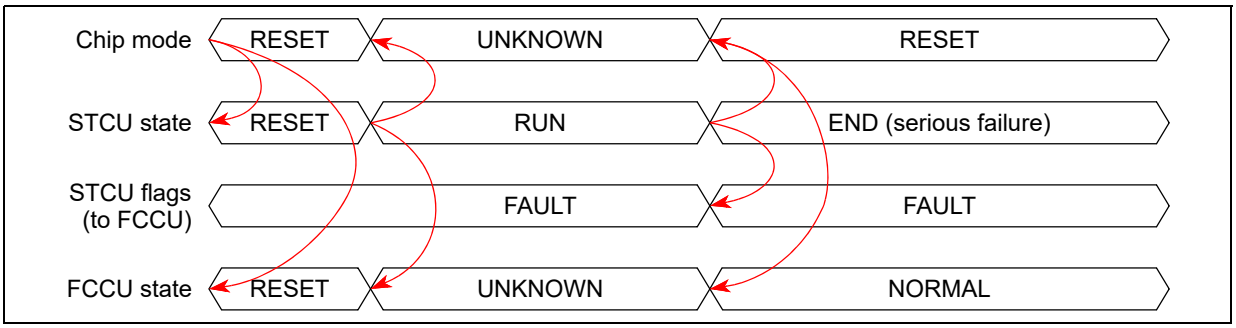


Figure 851. STCU-FCCU (case c)



51.5.9 EOUT interface

The FCCU provides up to two bidirectional signals (EOUT interface) as a failure indication to the external world (refer to table “FCCU failure inputs” in the “Functional safety” chapter for details). These signals need to be mapped to appropriate pins for external visibility. Since IO port does not control the pad buffer enable during Alternate function, FCCU provides a mechanism to drive the pad in OD (Open Drain) mode where the pad is driven Hi-Z when FCCU drives logic ‘1’ else it drives logic ‘0’ (when FCCU drives logic '0'). The selection to OD is done by OD bit in FCCU_CFG register. Different protocols for the EOUT interface are supported, selecting the FCCU_CFG.FOM register field:

- Dual rail protocol
- Time switching protocol
- Bi-stable protocol
- Test mode

The signal polarity can be programmed by setting the PS field in the FCCU_CFG register. For example, for PS = 0, FCCU Error output[0] = 0 and FCCU Error output[1] = 1 in FAULT state for bi-stable and for PS = 1, it is inverted. All the diagrams and tables are related to the default configuration selection: switching mode (FCCU_CFG.SM = 0b). In case of inverted polarity (FCCU_CFG.PS = 1b) all the values on the EOUT output pins are inverted.

Two modes can be programmed to define the EOUT protocol transitions in dual rail or time switching mode:

- Slow switching mode: no EOUT frequency violation during the FCCU state transition (NORMAL to FAULT or vice-versa and CONFIG to NORMAL). The EOUT protocol transition occurs after a max delay equal to the duration of the semi-period of the EOUT frequency.

- Fast switching mode: The EOUT protocol transition (NORMAL to FAULT or vice-versa and CONFIG to NORMAL) occurs immediately. A pulse with the minimum duration corresponding to 16 MHz/1024 (IRCOSC clock) period can occur in fast switching mode. It implies a frequency violation of the EOUT protocol.

The EOUT frequency is generated by dividing IRCOSC clock by a fixed factor of 2^{18}. With an IRCOSC clock of 16 MHz, this drives a signal of 61 Hz on EOUT.

The external monitor of the EOUT protocol must oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and the IRCOSC clock detecting the edge transition of the EOUT protocol in dual rail or time switching mode.

In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before. If SW configures the error pins to OK(1) and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and error pins indicate NOK, that is an incoming fault is not masked when SW has set the error pin to high. During the T_min by a non-SW fault, the FCCU FSM moves independently of this pin state (low) and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in and it is not possible to set the pins to OK by SW moving FCCU to CONFIG state, as long as this timer is running. No SW intervention is needed to bring the pin from the low state. The SW can bring the pin back to OK state by clearing the faults and waiting for the T_min to expire, after which the FCCU automatically enters NORMAL state and the error pin indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

Resets (except the power on reset) must not influence the state of the failure indication, pins.

Note: The transition from FAULT to CONFIG state is not possible but is shown to display the behavior in all four states: RESET, NORMAL, FAULT and CONFIG.

Table 580. FCCU error pin behavior on STATE transitions(1)

Fault signaling state	Transition from NORMAL to FAULT state			NORMAL+CONFIG states			Transition from NORMAL to ALARM state		
	with Fault disabled | enabled						with Fault disabled | enabled		
	Internal Error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA | OK	NA | 1	NA | OK	OK	1	OK	NA | OK	NA | 1	NA | OK
Enabled	NA | NOK	NA | 1	NA | NOK	OK	1	OK	NA | OK	NA | 1	NA | OK

1. Table valid only after SW has written FCCU_SET_AFTER_RESET bit to 1.

Note: FCCU remains in NORMAL state when fault is disabled hence, “NA” (Not Applicable) appears in the above table.

51.5.9.1 Dual rail protocol

Dual rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual rail encoded circuits use two

wires to carry each bit. The encoding scheme is given in *Table 581* and the related timing diagram is given in *Figure 852* and *Figure 853*.

Table 581. Dual rail encoding

Logical state	Dual rail encoding (FCCU Error output pins [1:0])	Note
Nonfaulty	10	Toggling
Nonfaulty	01	
Faulty	00	Toggling
Faulty	11	
Reset	High-z(1)	No toggling
Configuration	Same as NORMAL(2)	Toggling

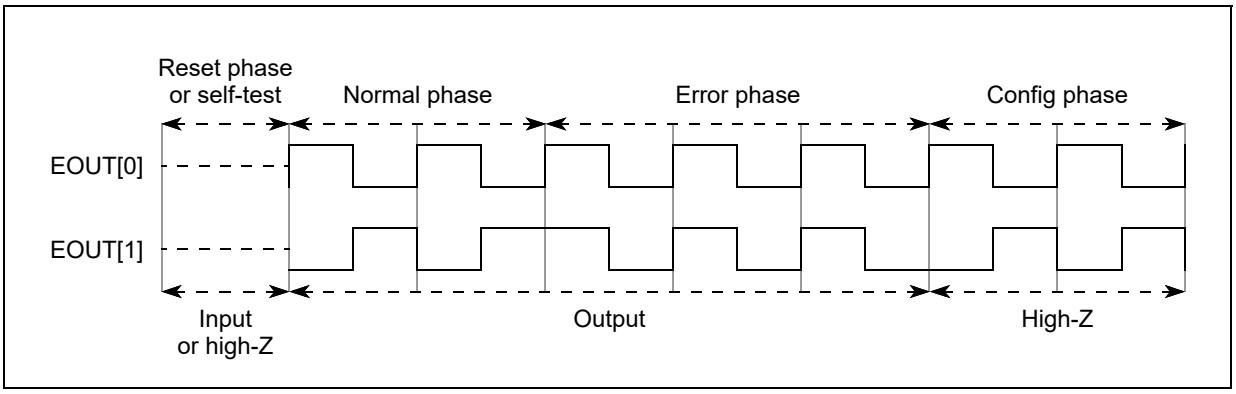
1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

As long as FCCU is in NORMAL or ALARM state, outputs show “nonfaulty” signal. Output pins #0, #1 toggle between 01 and 10.

In the RESET phase the output pins are set as “high impedance”(d).

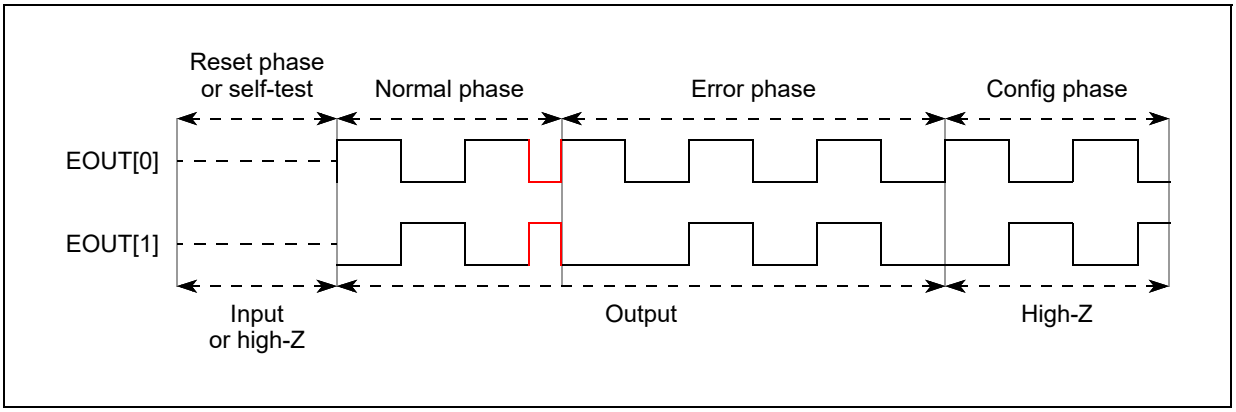
Note: Figure 852 and Figure 853 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

Figure 852. Dual rail protocol (slow switching mode)



d. Actual value depends on the SOC settings at pad level.

Figure 853. Dual rail protocol (fast switching mode)



51.5.9.2 Time switching protocol

The encoding scheme is given in *Table 582* and the related timing diagram is given in *Figure 854*.

Table 582. Time switching encoding

Logical state	Time switching encoding (FCCU Error output pins [1:0])	Note
Nonfaulty	10	Toggling
Nonfaulty	01	
Faulty	10	No toggling
Reset	High-Z(1)	No toggling
Configuration	Same as NORMAL(2)	Toggling

1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

As long as FCCU is in NORMAL or ALARM state, outputs show “nonfaulty” signal. Output pins #0, #1 toggle between 01 and 10.

In the FAULT state, the output pin FCCU Error output pin[0] is set as low.

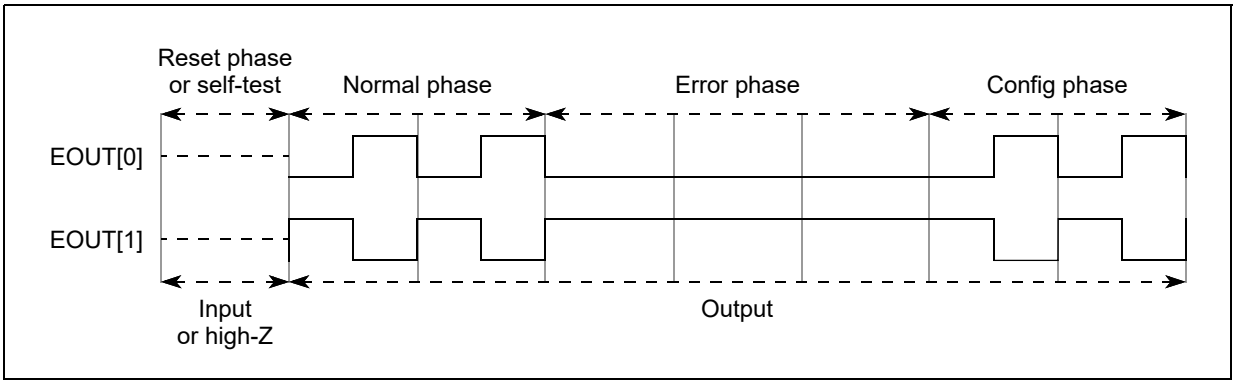
In Time Switching mode the second output (FCCU Error output pin[1]) is the inverted signal of first output (FCCU Error output pin[0]).

In the RESET phase the output pins are set as “high impedance”(e).

Note: Figure 854 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

e. Actual value depends on the SOC settings at pad level.

Figure 854. Time switching protocol



51.5.9.3 Bi-stable protocol

The encoding scheme is given in *Table 583* and the related timing diagram is given in *Figure 855*.

Table 583. Bi-stable encoding

Logical state	Bi-stable encoding (FCCU Error output pins [1:0])	Note
Nonfaulty	01	No toggling
Faulty	10	No toggling
Reset	High-z(1)	No toggling
Configuration	Same as Normal(2)	No toggling

1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

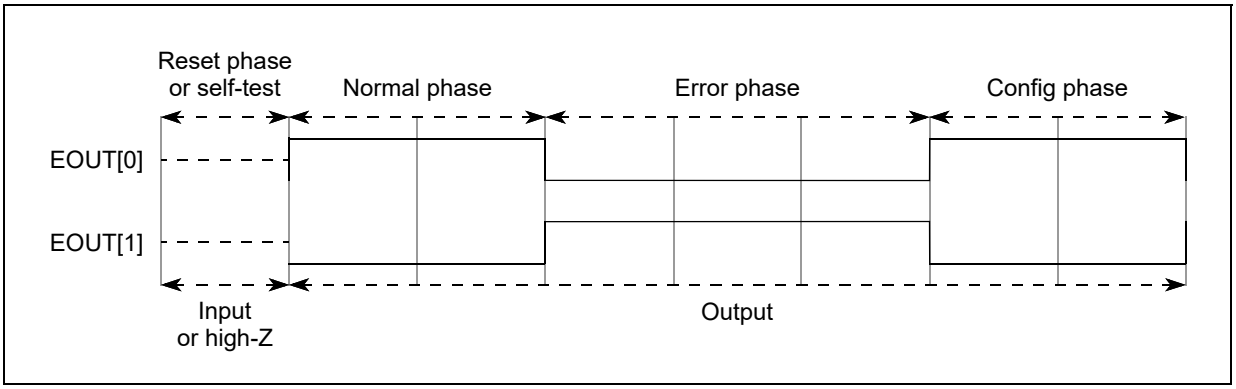
In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, “no-faulty” state is indicated. In Bi-stable mode the second output (FCCU Error output[1]) is the inverted signal of first output (FCCU Error output[0]).

In the RESET phase the output pins are set as “high impedance”(f).

Note: Figure 855 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

f. Actual value depends on the SOC settings at pad level.

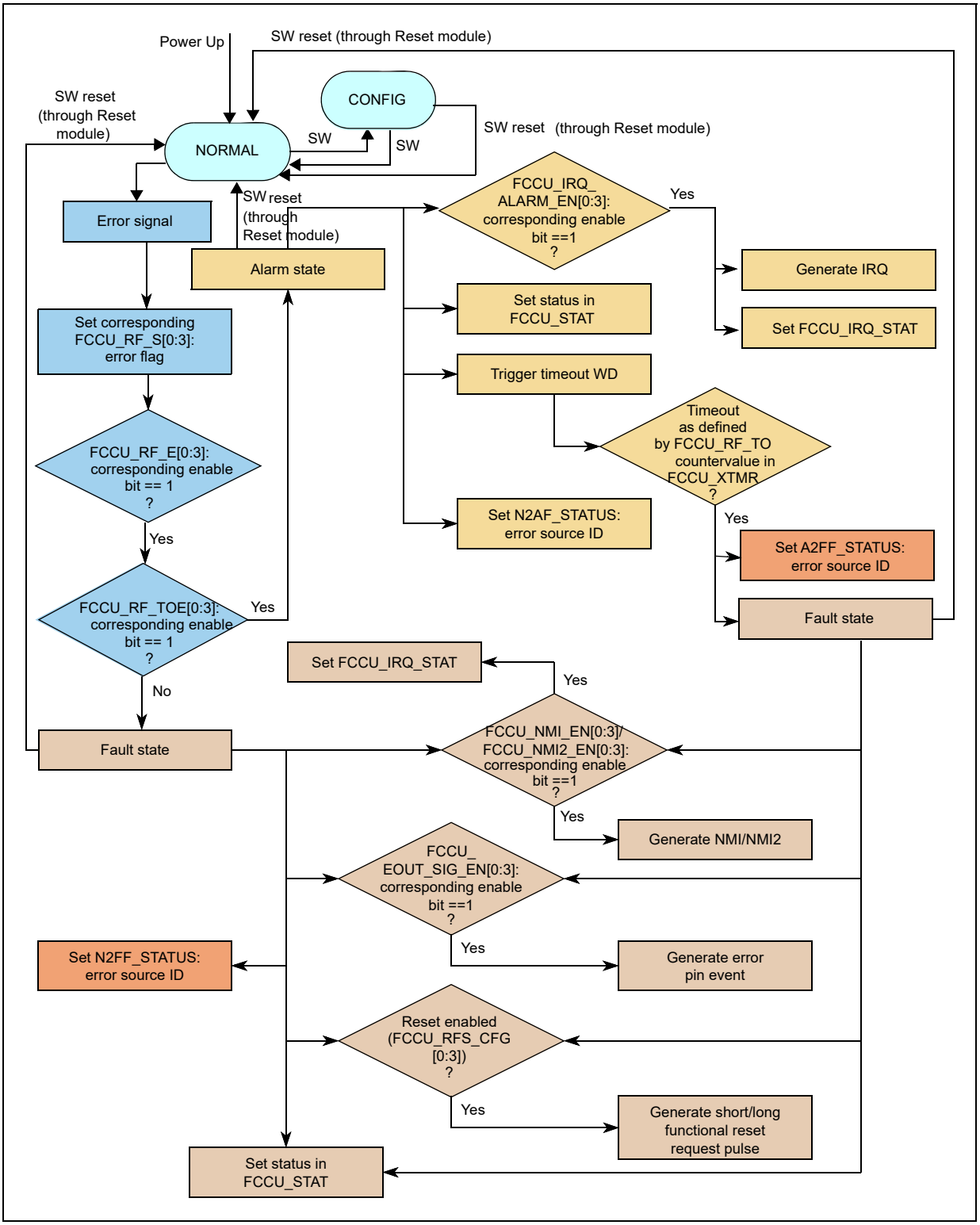
Figure 855. Bi-stable protocol



51.5.10 Error signal flow for RF case

On Power up, FCCU moves to NORMAL state. SW can configure it by moving it to CONFIG and back again. SW reset leads to movement back to NORMAL. On reaching ALARM state, depending on ALARM being enabled, FCCU generates IRQ (alarm) and the FCCU_IRQ_STAT's alarm status is also set. FSM status is set in FCCU_STAT along with N2AF_STATUS. If Error is enabled (FCCU_RF_E) and timeout disabled (FCCU_RF_TOE), FCCU moves straight to FAULT state and depending on FCCU_NMI_EN/FCCU_NMI2_EN or FCCU_SIG_EN, or both, being set, it generates NMIs or signals error, or both, out to the external world. FCCU FSM status is also captured in FCCU_STAT.

Figure 856. Error signal flow



51.6 Register description

The FCCU registers are listed in *Table 584*. Any address offset not explicitly mentioned in *Table 584* is reserved.

All the registers accessible in write mode only in CONFIG state (last column of the *Table 584*) are referred to as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible RF failure source a different reaction must be configurable through the use of NMIs, IRQ, long/short reset selection registers as well as no reaction by disabling the former registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it is possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

Table 584. FCCU memory map

Address offset	Register	Access(1)	Location
0x000	FCCU Control Register (FCCU_CTRL)	R/W always	*Section 51.6.1*
0x004	FCCU CTRL Key Register (FCCU_CTRLK)	W always	*Section 51.6.2*
0x008	FCCU Configuration Register (FCCU_CFG)	R always; W in CONFIG state only	*Section 51.6.3*
0x01C	FCCU RF Configuration Register 0 (FCCU_RF_CFG0)	R always; W in CONFIG state only	*Section 51.6.4*
0x020	FCCU RF Configuration Register 1 (FCCU_RF_CFG1)		
0x024	FCCU RF Configuration Register 2 (FCCU_RF_CFG2)		
0x04C	FCCU RFS Configuration Register 0 (FCCU_RFS_CFG0)	R always; W in CONFIG state only	*Section 51.6.5*
0x050	FCCU RFS Configuration Register 1 (FCCU_RFS_CFG1)		
0x054	FCCU RFS Configuration Register 2 (FCCU_RFS_CFG2)		
0x058	FCCU RFS Configuration Register 3 (FCCU_RFS_CFG3)		
0x05C	FCCU RFS Configuration Register 4 (FCCU_RFS_CFG4)		
0x060	FCCU RFS Configuration Register 5 (FCCU_RFS_CFG5)		
0x080	FCCU RF Status Register 0 (FCCU_RF_S0)	R/W always	*Section 51.6.6*
0x084	FCCU RF Status Register 1 (FCCU_RF_S1)		
0x088	FCCU RF Status Register 2 (FCCU_RF_S2)		
0x090	FCCU RF Key Register (FCCU_RFK)	W always	*Section 51.6.7*
0x094	FCCU RF Enable Register 0 (FCCU_RF_E0)	R always; W in CONFIG state only	*Section 51.6.8*
0x098	FCCU RF Enable Register 1 (FCCU_RF_E1)		
0x09C	FCCU RF Enable Register 2 (FCCU_RF_E2)		

Table 584. FCCU memory map (continued)

Address offset	Register	Access[1]	Location
0x0A4	FCCU RF Timeout Enable Register 0 (FCCU_RF_TOE0)	R always; W in CONFIG state only	*Section 51.6.9*
0x0A8	FCCU RF Timeout Enable Register 1 (FCCU_RF_TOE1)		
0x0AC	FCCU RF Timeout Enable Register 2 (FCCU_RF_TOE2)		
0x0B4	FCCU RF Timeout Register (FCCU_RF_TO)	R always; W in CONFIG state only	*Section 51.6.10*
0x0B8	FCCU CFG Timeout Register (FCCU_CFG_TO)	R always; W in all states except CONFIG	*Section 51.6.11*
0x0BC	FCCU IO Control Register (FCCU_EINOUT)	R/W always	*Section 51.6.12*
0x0C0	FCCU Status Register (FCCU_STAT)	R/W always	*Section 51.6.13*
0x0C4	FCCU NA Freeze Status Register (N2AF_STATUS)	R/W always	*Section 51.6.14*
0x0C8	FCCU AF Freeze Status Register (A2FF_STATUS)	R/W always	*Section 51.6.15*
0x0CC	FCCU NF Freeze Status Register (N2FF_STATUS)	R/W always	*Section 51.6.16*
0x0D0	FCCU FA Freeze Status Register (F2A_STATUS)	R/W always	*Section 51.6.17*
0x0DC	FCCU RF Fake Register (FCCU_RFF)	R/W always	*Section 51.6.18*
0x0E0	FCCU IRQ Status Register (FCCU_IRQ_STAT)	R/W always	*Section 51.6.19*
0x0E4	FCCU IRQ Enable Register (FCCU_IRQ_EN)	R/W always	*Section 51.6.20*
0x0E8	FCCU XTMR Register (FCCU_XTMR)	R/W always	*Section 51.6.21*
0x0F0	FCCU Transient Lock Register (FCCU_TRANS_LOCK)	W always	*Section 51.6.22*
0x0F4	FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)	W always	*Section 51.6.23*
0x0F8	FCCU Delta T Register (FCCU_DELTA_T)	R always; W in CONFIG state only	*Section 51.6.24*
0x0FC	FCCU IRQ Alarm Enable Register 0 (FCCU_IRQ_ALARM_EN0)	R always; W in CONFIG state only	*Section 51.6.25*
0x100	FCCU IRQ Alarm Enable Register 1 (FCCU_IRQ_ALARM_EN1)		
0x104	FCCU IRQ Alarm Enable Register 2 (FCCU_IRQ_ALARM_EN2)		
0x10C	FCCU NMI Enable Register 0 (FCCU_NMI_EN0)	R always; W in CONFIG state only	*Section 51.6.26*
0x110	FCCU NMI Enable Register 1 (FCCU_NMI_EN1)		
0x114	FCCU NMI Enable Register 2 (FCCU_NMI_EN2)		

Table 584. FCCU memory map (continued)

Address offset	Register	Access(1)	Location
0x11C	FCCU EOUT Signaling Enable Register 0 (FCCU_EOUT_SIG_EN0)	R always; W in CONFIG state only	*Section 51.6.27*
0x120	FCCU EOUT Signaling Enable Register 1 (FCCU_EOUT_SIG_EN1)		
0x124	FCCU EOUT Signaling Enable Register 2 (FCCU_EOUT_SIG_EN2)		
0x12C	FCCU NMI2 Enable Register 0 (FCCU_NMI2_EN0)	R always; W in CONFIG state only	*Section 51.6.28*
0x130	FCCU NMI2 Enable Register 1 (FCCU_NMI2_EN1)		
0x134	FCCU NMI2 Enable Register 2 (FCCU_NMI2_EN2)		

1. In this column, R/W = read/write, R = read-only, and W = write-only.

51.6.1 FCCU control register (FCCU_CTRL)

The FCCU_CTRL register allows to execute the following operations:

- moving the FCCU state from the NORMAL state into the CONFIG state.
- moving the FCCU state from the CONFIG state into the NORMAL state.
- reading or to clear the RF status register.
- reading the FCCU FSM status register.
- reading or to clear the FCCU freeze registers.
- reading the ALARM timer.
- reading the Watchdog timer.

Some critical operations require a key as defined in the FCCU_CTRLK register.

Offset: 0x000 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DEBUG	0	OPS		0	OPR				
W																
Reset	0	0	0	0	0	0	0	0	0/1(1)	0/1(1)	0	0	0	0	0	0

1. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 857. FCCU Control Register (FCCU_CTRL)

Table 585. FCCU_CTRL field descriptions

Field	Description
9 DEBUG	Debug mode entry 0 Normal operation 1 Put FCCU into debug mode When Debug pin signal is asserted and this bit is set, FCCU moves into debug state and suspends the transition of the FSM by suspending timer elapse time effect.
7:6 OPS	Operation status 00 Idle 01 In progress 10 Aborted 11 Successful These bits can be read and cleared (via OP15 operation) by the software.

Table 585. FCCU_CTRL field descriptions (continued)

Field	Description
4:0 OPR	Operation run 00000 No operation [OP0] 00001 Set the FCCU into the CONFIG state [OP1] 00010 Set the FCCU into the NORMAL state [OP2] 00011 Read the FCCU state (refer to the FCCU_STAT register) [OP3] 00100 Read the FCCU frozen status flags (refer to the N2AF_STATUS register) [OP4] 00101 Read the FCCU frozen status flags (refer to the A2FF_STATUS register) [OP5] 00110 Read the FCCU frozen status flags (refer to the N2FF_STATUS register) [OP6] 00111 Read the FCCU frozen status flags (refer to the F2A_STATUS register) [OP7] 01000 Reserved 01001 Reserved 01010 Read the RF status register (refer to the FCCU_RF_S register) [OP10] 01011 Reserved 01100 RF status clear operation in progress (refer to the FCCU_RF_S register) [OP12] 01101 Clear the freeze status registers (refer to the freeze registers) [OP13] 01110 CONFIG to NORMAL FCCU state (configuration timeout) in progress [OP14] 01111 Clear the operation status (OPS=Idle) [OP15] 10000 Reserved 10001 Read the ALARM timer (refer to the FCCU_XTMR register) [OP17] 10010 RESERVED 10011 Read the CFG timer (refer to the FCCU_XTMR register) [OP19] 10100 Read the Error Pin low counter value (refer to *Section 51.6.21: FCCU XTMR register (FCCU_XTMR)*) [OP20] 10101 Reserved ... 11111 Reserved The SW application must not modify the OPR field (any write operation is ignored) until the completion of the operation. Once the operation has been completed, the operation status (OPS) is set and the OPR field is automatically cleared (OPR = 000). The opcode OP11, OP12 must not be programmed. The OPR field is automatically set to OP11 or OP12 when the FCCU_RF_S registers are cleared by a write-clear operation into the related register. The opcode OP14 must not be programmed. The OPR field is automatically set to OP14 when the timeout occurs (FCCU_CFG_TO) during the configuration procedure => the FCCU state is automatically forced in NORMAL mode setting the default configuration. In this phase any write operation to the FCCU configuration registers is inhibited. The reserved operations (OP21 -OP30) are forbidden. In case of execution, they return an ABORT response without any side effect. The ABORT response occurs in the following cases: – wrong access (missing or wrong key) to the FCCU_RF_S register (clear operation OP12) – wrong access (missing or wrong key) to the FCCU_CTRL register (OP1, OP2 operation) – OP1 (CONFIG command) execution when FCCU state /= NORMAL or configuration locked – OP21-OP30 (RESERVED operations) execution The max timing needed for ABORT/SUCCESS response is 15 IRC cycles

51.6.2 FCCU CTRL key register (FCCU_CTRLK)

The FCCU_CTRLK register implements the key access for the operations OP1, OP2 according to the following sequence:

1. Write the key into the FCCU_CTRLK register.
2. Write the FCCU_CTRL register (operations OP1 or OP2).

Note: There must not be any other operation in between the above steps and the 32 bits need to be written in one cycle. So, stores FCCU_CTRLK, merges with OP1 or OP2 then sends it.

The FCCU_CTRLK register is not readable, a 0x0000_0000 value is always returned in case of read operation. The key must be written by a word (32 bits) data write operation.

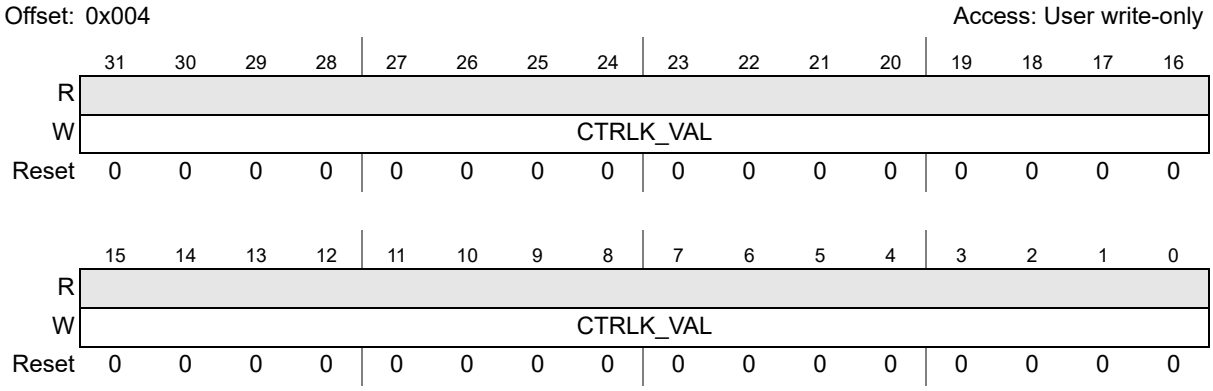


Figure 858. FCCU CTRL Key Register (FCCU_CTRLK)

Table 586. FCCU_CTRLK field descriptions

Field	Description
31:0 CTRLK_VAL	Control register key 0x913756AF Key for the operation OP1 0x825A132B Key for the operation OP2

51.6.3 FCCU configuration register (FCCU_CFG)

The FCCU_CFG register defines the global configuration for the FCCU module.

This register is writable only in the CONFIG state.

Offset: 0x008 Access: User read/write(1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							FCCU_SET_AFTER_RESET	FCCU_SET_CLEAR		0	0	Reserved			
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	Reserved		OD	0	SM	PS	FOM			Reserved					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0(2)	0	0	0

1. Writable only in the CONFIG state.
2. After power up, reset value is 0. If configuration timeout occurs then the default value is changed to 1.

Figure 859. FCCU Configuration Register (FCCU_CFG)

Table 587. FCCU_CFG field descriptions

Field	Description
24 FCCU_SET_AFTER_RESET	This bit controls the enable of the o/p error pin after reset lifts 0 FCCU's Error indication stops functioning and error pins are in HI-Z state(1). 1 write to start FCCU functioning. After power-on the error out pins must be in high impedance(1). They go to normal state only on software request, this bit is set to 1. It is the software responsibility to write this bit to 1 else the error pin stays in High-Z state after reset lifts and FCCU is not functioning.
23:22 FCCU_SET_CLEAR	Error pin state can be controlled by these bits 01 Error Pin driven to logic 0. 11 Error Pin driven to logic 1 (write not possible when FCCU already in the T min timer phase or if FSM enters in FAULT state by capture of a fault). 00 FCCU acts independent of above SW control. 10 FCCU acts independent of above SW control. These bits clear(0) and set(1) the error pin. Higher priority is of the FCCU_SET_AFTER_RESET bit's capability to lead the error pins to Hi-Z(1). Switch to software configuration (01, 11) comes into effect when FCCU FSM leaves CONFIG state. After this is in effect, software maintains priority over FCCU FSM except for the case of '11' mentioned above. The switch back to FSM control is again done by writing into these fields with 00 or 10 by going to CONFIG state. This bit feature may not work with toggling protocols for example say EOUT pins are set high by SW using this bit followed by switching to a toggling protocol after returning control to FCCU FSM via setting this bit to 00 or 10, the user may find that EOUT pins do not toggle as was being expected.

Table 587. FCCU_CFG field descriptions (continued)

Field	Description
12 OD	Open Drain Mechanism to select between Push-pull and Open drain (OD) mode for the error indicating pin(s). 0 push-pull 1 OD
10 SM	Switching mode 0 EOUT protocol (dual rail, time switching) slow switching mode. 1 EOUT protocol (dual rail, time switching) fast switching mode. SM has no effect on the bi-stable protocol.
9 PS	Polarity selection 0 FCCU Error output[1] active high, FCCU Error output[0] active low. 1 FCCU Error output[1] active low, FCCU Error output[0] active high. This bit can be read and written by the software. Active state here means a state indicating FAULT and applicable when pins are non toggling in Error phase.
8:6 FOM	Fault Output Mode selection 000 Dual Rail (default state) [FCCU Error output[1:0]= outputs] 001 Time Switching [FCCU Error output[1:0]= output to be used] 010 Bi-Stable 011 Reserved 100 Reserved 101 Test0 (controlled by FCCU_EOUT register) [FCCU Error output[0]= input, FCCU Error output[1]= output] 110 Test1 (controlled by FCCU_EOUT register) [FCCU Error output[0]= output, FCCU Error output[1]= output] 111 Test2 (controlled by FCCU_EOUT register) [FCCU Error output[0]= output, FCCU Error output[1]= input] **Note:** In Test mode, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs on the system/IRCOSC clock.

1. Actual value depends on the SOC settings at pad level.

51.6.4 FCCU RF configuration register *n* (FCCU_RF_CFG*n*)

The FCCU_RF_CFG*n* registers contain the configuration of each recoverable fault in terms of fault recovery management. The configuration depends on the type of signaling of a fault event. HW recoverable faults must be configured only if a previous latching stage captures and holds the physical fault otherwise the fault can be lost. All the other faults must be configured as SW fault.

These registers are writable only in the CONFIG state.

Offset: 0x01C + *n**0x4 (*n* = 0 to 2) Access: User read/write(1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RFC31(2)	RFC30(2)	RFC29(2)	RFC28(2)	RFC27(2)	RFC26(2)	RFC25(2)	RFC24(2)	RFC23(2)	RFC22(2)	RFC21(2)	RFC20(2)	RFC19(2)	RFC18(2)	RFC17(2)	RFC16(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RFC15(2)	RFC14(2)	RFC13(2)	RFC12(2)	RFC11(2)	RFC10(2)	RFC9(2)	RFC8(2)	RFC7(2)	RFC6(2)	RFC5(2)	RFC4(2)	RFC3(2)	RFC2(2)	RFC1(2)	RFC0(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 589* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 860. FCCU RF Configuration Register *n* (FCCU_RF_CFG*n*)

Table 588. FCCU_RF_CFG*n* field descriptions

Field	Description
31:0 RFC[31:0]	Recoverable fault configuration 0 HW recoverable fault 1 SW recoverable fault The recoverable fault configuration defines the fault recovery mode. HW recoverable faults are self recovered (status flag clearing and related) if the root cause (input fault) has been removed. SW recoverable faults are recovered (status flag clearing) by SW clearing of the related status flag.

Table 589: FCCU RF configuration register channels shows FCCU RF configuration register channels.

Table 589. FCCU RF configuration register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x01C	FCCU_RF_CFG0	RFC[31:0]
0x020	FCCU_RF_CFG1	RFC[63:32]
0x024	FCCU_RF_CFG2	RFC[95:64]

51.6.5 FCCU RFS configuration register *n* (FCCU_RFS_CFG*n*)

The FCCU_RFS_CFG*n* registers contain the configuration of each recoverable fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

These registers are writable only in the CONFIG state.

Offset: 0x04C +*n**0x4 (*n* = 0 to 5) Access: User read/write(1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RFSC15(2)		RFSC14(2)		RFSC13(2)		RFSC12(2)		RFSC11(2)		RFSC10(2)		RFSC9(2)		RFSC8(2)	
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RFSC7(2)		RFSC6(2)		RFSC5(2)		RFSC4(2)		RFSC3(2)		RFSC2(2)		RFSC1(2)		RFSC0(2)	
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 591* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 861. FCCU RFS Configuration Register *n* (FCCU_RFS_CFG*n*)

Table 590. FCCU_RFS_CFG*n* field descriptions

Field	Description
31:0 RFSC[15:0]	Recoverable fault state configuration 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction) 10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

Table 591: FCCU RFS configuration register channels shows FCCU **RFS** configuration register channels.

Table 591. FCCU RFS configuration register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x04C	FCCU_RFS_CFG0	RFSC[15:0]
0x050	FCCU_RFS_CFG1	RFSC[31:16]
0x054	FCCU_RFS_CFG2	RFSC[47:32]
0x058	FCCU_RFS_CFG3	RFSC[63:48]
0x05C	FCCU_RFS_CFG4	RFSC[79:64]
0x060	FCCU_RFS_CFG5	RFSC[95:80]

51.6.6 FCCU RF status register *n* (FCCU_RF_S*n*)

The FCCU_RF_S*n* register contains the latched fault indication collected from the recoverable fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the RF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_RF_E*n* register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_RF_E*n* register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before recovering the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_RF_S*n* register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_RF_S*n* register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_RFK register
2. Clear the status (flag) bit RFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field
3. Wait for the completion of the operation (FCCU_CTRL.OPS field)
4. Read the FCCU_RF_S*n* register in order to verify the effective deletion and in case of failure to repeat the sequence

As a result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root (refer to *Section 51.5.7: FAULT interface*).

Note: There must not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU remains in the FAULT or ALARM state.

The SW application executes the FCCU_RF_S*n* read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the FCCU_RF_S*n* register

In case of reconfiguration of the FCCU (CONFIG state), before returning in NORMAL state the pending status bits into the FCCU_RF_S*n* must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following errors are ignored:

- To write a wrong key into the FCCU_RFK register
- To attempt to clear a HW recoverable error

Offset: 0x080 + *n**0x4 (*n* = 0 to 2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RFS31(1)	RFS30(1)	RFS29(1)	RFS28(1)	RFS27(1)	RFS26(1)	RFS25(1)	RFS24(1)	RFS23(1)	RFS22(1)	RFS21(1)	RFS20(1)	RFS19(1)	RFS18(1)	RFS17(1)	RFS16(1)
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset:	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RFS15(1)	RFS14(1)	RFS13(1)	RFS12(1)	RFS11(1)	RFS10(1)	RFS9(1)	RFS8(1)	RFS7(1)	RFS6(1)	RFS5(1)	RFS4(1)	RFS3(1)	RFS2(1)	RFS1(1)	RFS0(1)
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset:	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)

1. Refer to *Table 593* for register offset to channel number relationship.
2. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 862. FCCU RF Status register *n* (FCCU_RF_S*n*)

Table 592. FCCU_RF_S*n* field descriptions

Field	Description
31:0 RFS[31:0]	Recoverable fault status 0 No recoverable fault latched 1 Recoverable fault latched The status bits related to the recoverable fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed.

Table 593: FCCU RF status register channels shows FCCU **RF** status register channels.

Table 593. FCCU RF status register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x080	FCCU_RF_S0	RFS[31:0]
0x084	FCCU_RF_S1	RFS[63:32]
0x088	FCCU_RF_S2	RFS[95:64]

51.6.7 FCCU RF key register (FCCU_RFK)

The FCCU_RFK register implements the key access to clear the status flags of the FCCU_RF_S*n* register.

The status bits of the FCCU_RF_S*n* register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_RFK register
2. Clear the status (flag) bit RFSx

The key must be written for each FCCU_RF_S*n* clear operation.

Note: There must not be any other operation in between the above steps.

The FCCU_RFK register is not readable; a 0x0000_0000 value is always returned in case of read operation.

Offset: 0x090 Access: User Write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	RFK_VAL															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	RFK_VAL															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 863. FCCU RF Key register (FCCU_RFK)

Table 594. FCCU_RFK field descriptions

Field	Description
31:0 RFK_VAL	recoverable fault key = 0xAB34_98FE

51.6.8 FCCU RF enable register *n* (FCCU_RF_E*n*)

The FCCU_RF_E*n* registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_RF_S*n* register is anyway set, only the reaction is masked.

These registers are writable only in the CONFIG state.

Offset: 0x094 + *n**0x4 (*n* = 0 to 2) Access: User read/write(1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RFE31(2)	RFE30(2)	RFE29(2)	RFE28(2)	RFE27(2)	RFE26(2)	RFE25(2)	RFE24(2)	RFE23(2)	RFE22(2)	RFE21(2)	RFE20(2)	RFE19(2)	RFE18(2)	RFE17(2)	RFE16(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RFE15(2)	RFE14(2)	RFE13(2)	RFE12(2)	RFE11(2)	RFE10(2)	RFE9(2)	RFE8(2)	RFE7(2)	RFE6(2)	RFE5(2)	RFE4(2)	RFE3(2)	RFE2(2)	RFE1(2)	RFE0(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 596* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 864. FCCU RF Enable Register *n* (FCCU_RF_E*n*)

Table 595. FCCU_RF_E*n* field descriptions

Field	Description
31:0 RFE[31:0]	Recoverable fault enable 0 No actions following the respective recoverable fault assertion 1 FCCU moves to ALARM or FAULT state

Table 596: FCCU RF_En enable register channels shows FCCU **RF_E*n*** enable register channels.

Table 596. FCCU RF_E*n* enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x094	FCCU_RF_E0	RFE[31:0]
0x098	FCCU_RF_E1	RFE[63:32]
0x09C	FCCU_RF_E2	RFE[95:64]

51.6.9 FCCU RF timeout enable register *n* (FCCU_RF_TOE*n*)

The FCCU_RF_TOE*n* registers enable a transition from the NORMAL state into the ALARM state if the respective recoverable fault is enabled (RFEx and RFTOEx are set). In case the respective timeout is disabled (RFTOEx is cleared) and the recoverable fault is enabled (RFEx is set) the FCCU moves into the FAULT state if the related recoverable fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

These registers are writable only in the CONFIG state.

Offset: 0x0A4 + *n**0x4 (*n* = 0 to 2) Access: User read/write(1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RFTOE31(2)	RFTOE30(2)	RFTOE29(2)	RFTOE28(2)	RFTOE27(2)	RFTOE26(2)	RFTOE25(2)	RFTOE24(2)	RFTOE23(2)	RFTOE22(2)	RFTOE21(2)	RFTOE20(2)	RFTOE19(2)	RFTOE18(2)	RFTOE17(2)	RFTOE16(2)
W																
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RFTOE15(2)	RFTOE14(2)	RFTOE13(2)	RFTOE12(2)	RFTOE11(2)	RFTOE10(2)	RFTOE9(2)	RFTOE8(2)	RFTOE7(2)	RFTOE6(2)	RFTOE5(2)	RFTOE4(2)	RFTOE3(2)	RFTOE2(2)	RFTOE1(2)	RFTOE0(2)
W																
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 598* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 865. FCCU RF Timeout Enable Register *n* (FCCU_RF_TOE*n*)

Table 597. FCCU_RF_TOE*n* field descriptions

Field	Description
31:0 RFTOE[31:0]	Fault timeout enable 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled.

Table 598: FCCU RF timeout enable register channels shows FCCU **RF** timeout enable register channels.

Table 598. FCCU RF timeout enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x0A4	FCCU_RF_TOE0	RFTOE[31:0]
0x0A8	FCCU_RF_TOE1	RFTOE[63:32]
0x0AC	FCCU_RF_TOE2	RFTOE[95:64]

51.6.10 FCCU RF timeout register (FCCU_RF_TO)

The FCCU_RF_TO register defines the preset value of the timer for the recovery of the recoverable faults (enabled). Once FCCU enters in ALARM state, following the assertion of a recoverable fault enabled (RFEx and RFTOEx are set), the timer starts the count down.

If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state. The alarm timeouts value must be programmed less than the FOSU_COUNT, else destructive resets may be generated by FOSU timeout.

This register is writable only in the CONFIG state.

Offset: 0x0B4 Access: User read/write(1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TO															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TO															
Reset:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. Writable only in the CONFIG state.

Figure 866. FCCU RF Timeout Register (FCCU_RF_TO)

Table 599. FCCU_RF_TO field descriptions

Field	Description
31:0 TO	Recoverable fault timeout Timeout = (TO) x $T_{RC16MHz}$

51.6.11 FCCU CFG timeout register (FCCU_CFG_TO)

The FCCU_CFG_TO register defines the preset value of the watchdog timer for the recovery from the CONFIG state. Once FCCU enters in CONFIG state, following a SW request (OP1 opcode), the watchdog timer is initialized and starts the countdown if the reset is not asserted.

If the configuration is not completed within the timeout, the FCCU moves automatically from the CONFIG state to the NORMAL state and the default values for all the configuration register is restored. Configuration registers are those registers which can be written only in CONFIG state. Refer to *Table 584: FCCU memory map*. The watchdog timeout is clocked with the RC oscillator clock (16 MHz). The default timeout value is 4.096 ms. Longer activation of CONFIG state can lead to resets if a failure is indicated during the time the FCCU is in CONFIG state due to FOSU.

The FCCU_CFG_TO register is writable in any state excluded the CONFIG state as it follows the execution of the OP1 opcode (NORMAL to CONFIG state) and until the completion of the OP2 opcode (CONFIG to NORMAL state).

In case of watchdog timeout the FCCU_CFG_TO register is not accessible until the OP14 operation (CONFIG to NORMAL) has been completed.

Offset: 0x0B8 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TO		
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 867. FCCU CFG Timeout Register (FCCU_CFG_TO)

Table 600 shows FCCU_CFG_TO field descriptions.

Table 600. FCCU_CFG_TO field descriptions

Field	Description
2:0 TO	Configuration timeout Timeout = $T_{RC16MHz}$ x $2^{(TO + 10)}$ 000 Timeout = 64 µs ... 111 Timeout = 8.192 ms

51.6.12 FCCU IO control register (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- To control the FCCU Error output pin[1] output level when the FCCU is configured in “Test1” or “Test0” fault output mode (FCCU_CFG.FOM)
- To control the FCCU Error output pin[0] output level when the FCCU is configured in “Test1” or “Test2” fault output mode (FCCU_CFG.FOM)
- To observe the FCCU Error inputs level

Table 601: Bi-stable encoding shows Bi-stable encoding.

Table 601. Bi-stable encoding

Mode = FCCU_CFG.FOM	FCCU_EOUT[0]	FCCU_EOUT[1]
Test1	Output	Output
Test2	Output	Input
Test0	Input	Output

Note: Due to the resynchronization stage of the EOUT interface, there is a latency of a few IRCOSC clock cycles following a write/read operation of the FCCU_EINOUT register.

Offset: 0x0BC Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	EIN1(1)	EIN0(1)	0	0	EOUT1	EOUT0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0/1(2)	0/1(2)	0	0	0	0

1. FCCU Error input x depends on the PAD configuration, that is, internal pull-up enabled as default or pull-down and therefore EINx bit is 1 or 0 upon reset.
2. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 868. FCCU IO Control Register (FCCU_EINOUT)

Table 602. FCCU_EINOUT field descriptions

Field	Description
5 EIN1	Error input 1 0 when FCCU Error input[1] = 0 1 when FCCU Error input[1] = 1 The EIN1 captures the respective FCCU Error input[1] signal.
4 EIN0	Error input 0 0 when FCCU Error input[0] = 0 1 when FCCU Error input[0] = 1 The EIN0 captures the FCCU Error input[0] signal.
1 EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => FCCU Error output[1] configured in output mode). 0 force FCCU Error output[1] = 0 1 force FCCU Error output[1] = 1 The EOUT1 sets/clears the respective FCCU Error output[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a “do not care” value.
0 EOUT0	Error out 0 (significant only if the FCCU_CFG.FOM = Test1 or Test2 => FCCU Error output[0] configured in output mode) 0 force FCCU Error output[0] = 0 1 force FCCU Error output[0] = 1 The EOUT0 sets/clears the respective FCCU Error output[0] signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a “do not care” value.

51.6.13 FCCU status register (FCCU_STAT)

The FCCU_STAT register includes the FCCU status. The FCCU finite state machine (FSM) operates by the RC oscillator clock asynchronous with the System clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization

sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Offset: 0x0C0 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	PhysicError Pin		ESTAT	STATUS		
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 869. FCCU Status Register (FCCU_STAT)

Table 603. FCCU_STAT field descriptions

Field	Description
5:4 PhysicErrorPin	Physical Error Pin state PhysicErrorPin[5] corresponds to error pin 0, PhysicErrorPin[4] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0 Error Pin is at logical 0 1 Error Pin is at logical 1 During Fault state, a static or toggling value is observed based on selected protocol. **Note:** This field indicates the real time status of the Error pin as driven by FCCU (at FCCU boundary) to the pad and not what is seen at/from the pad, which is based on different IPs (such as IO port) configurations of the SoC.
3 ESTAT	Current Error Pin Status SW can write this bit with a different value. The new value is valid for 1 clock cycle, thereafter it returns to indicate the actual value being driven from FCCU. For example in nonfaulty states, this bit reads 0 and can be set for one cycle to 1. Similarly in faulty state, this bit reads 1 and sw can write it to 0 and after one cycle the bit reads 1 again. 0 System is OPERATIONAL (OK). 1 System is in FAULT state.This state is taken from the FCCU Eout logic (That is, as late as possible).
2:0 STATUS	FCCU Status 000 NORMAL state 001 CONFIG state 010 ALARM state 011 FAULT state Other: UNKNOWN state

51.6.14 FCCU NA freeze status register (N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the recoverable source (FCCU recoverable fault x) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified.

The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0C4 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	NAFS							
W									w1c							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 870. FCCU NA Freeze Status Register (N2AF_STATUS)

Table 604. N2AF_STATUS field descriptions

Field	Description
7:0 NAFS	Normal to Alarm Frozen Status 0x00 No transition from NORMAL to ALARM state 0x01 NORMAL to ALARM state transition cause => FCCU recoverable fault[0] 0x02 NORMAL to ALARM state transition cause => FCCU recoverable fault[1] 0x03 NORMAL to ALARM state transition cause => FCCU recoverable fault[2] ... 0x80 NORMAL to ALARM state transition cause => FCCU recoverable fault[127] 0xFF NORMAL to ALARM state transition cause => multiple FCCU recoverable faults

51.6.15 FCCU AF freeze status register (A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (recoverable fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0C8 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	AF_SRC		AFFS							
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 871. FCCU AF Freeze Status Register (A2FF_STATUS)

Table 605. A2FF_STATUS field descriptions

Field	Description
9:8 AF_SRC	Fault source 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple or recoverable, or both, faults
7:0 AFFS	Alarm to Fault Frozen Status 0x00 No transition from ALARM to FAULT state 0x01 ALARM to FAULT state transition cause => FCCU recoverable fault[0] timeout 0x02 ALARM to FAULT state transition cause => FCCU recoverable fault[1] timeout 0x03 ALARM to FAULT state transition cause => FCCU recoverable fault[2] timeout ... 0x80 ALARM to FAULT state transition cause => FCCU recoverable fault[127] timeout 0xFF ALARM to FAULT state transition cause => multiple FCCU recoverable faults timeout

51.6.16 FCCU NF freeze status register (N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the recoverable fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple

fault conditions the fault source cannot be identified. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0CC Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	NF_SRC		NFFS							
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 872. FCCU NF Freeze Status Register (N2FF_STATUS)

Table 606. N2FF_STATUS field descriptions

Field	Description
9:8 NF_SRC	Fault source 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple and recoverable faults
7:0 NFFS	Normal to Fault Frozen Status 0x00 No transition from NORMAL to FAULT state 0x01 NORMAL to FAULT state transition cause => FCCU recoverable fault[0] 0x02 NORMAL to FAULT state transition cause => FCCU recoverable fault[1] 0x03 NORMAL to FAULT state transition cause => FCCU recoverable fault[2] ... 0x80 NORMAL to FAULT state transition cause => FCCU recoverable fault[127] 0xFF NORMAL to FAULT state transition cause => multiple FCCU recoverable faults

51.6.17 FCCU FA freeze status register (F2A_STATUS)

The F2A_STATUS register contains a unique code to identify the recoverable fault source (recoverable fault x) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified.

The SW application executes the F2A_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the N2FF_STATUS register

The SW application executes the F2A_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0D0 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	FAFS							
W									w1c							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 873. FCCU FA Freeze Status Register (F2A_STATUS)

Table 607. F2A_STATUS field descriptions

Field	Description
7:0 FAFS	Fault to Normal Frozen Status 0x00 No transition from FAULT to NORMAL state 0x01 FAULT to ALARM state transition cause => FCCU recoverable fault[0] 0x02 FAULT to ALARM state transition cause => FCCU recoverable fault[1] 0x03 FAULT to ALARM state transition cause => FCCU recoverable fault[2] ... 0x80 FAULT to ALARM state transition cause => FCCU recoverable fault[127] 0xFF FAULT to ALARM state transition cause => multiple FCCU recoverable faults

51.6.18 FCCU RF fake register (FCCU_RFF)

The FCCU_RFF register contains a unique code to set a recoverable fault in mutually exclusive mode by the external FAULT interface signal setting. It allows the SW emulation of the recoverable faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The fault injection mechanism is optional and requires a proper management as described in *Section 51.5.7: FAULT interface*. The reaction following a fake recoverable fault cannot be masked. The FCCU_RFF is a write-only register with a set of codes corresponding to each recoverable fault injection.

Offset: 0x0DC Access: User Write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0							
W										FRFC						
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 874. FCCU RF Fake Register (FCCU_RFF)

Table 608. FCCU_RFF field descriptions

Field	Description
6:0 FRFC	Fake recoverable fault code 0x00 Fake recoverable fault injection at recoverable fault source 0 0x01 Fake recoverable fault injectionat recoverable fault source 1 0x02 Fake recoverable fault injection at recoverable fault source 2 … 0x7F Fake recoverable fault injection at recoverable fault source 127 **Note:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results.

51.6.19 FCCU IRQ status register (FCCU_IRQ_STAT)

The FCCU_IRQ_STAT register defines the FCCU interrupt status register related to the following events:

- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The external interrupt line (Interrupt request) is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Offset: 0x0E0 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	Reserved	Reserved	NMI_STAT	ALRM_STAT	CFG_TO_STAT
W																w1c
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 875. FCCU IRQ Status Register (FCCU_IRQ_STAT)

Table 609. FCCU_IRQ_STAT field descriptions

Field	Description
2 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
1 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
0 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

51.6.20 FCCU IRQ enable register (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration timeout error

The external interrupt line “Interrupt request (miscellaneous conditions)” is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

Offset: 0x0E4 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CFG_TO_IEN
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 876. FCCU IRQ Enable Register (FCCU_IRQ_EN)

Table 610. FCCU_IRQ_EN field descriptions

Field	Description
0 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled. 1 Configuration timeout interrupt enabled.

51.6.21 FCCU XTMR register (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer or EOUT pin low counter. These timers are clocked on the IRCOSC clock.

EOUTMR is the EOUT pin in logic 0 counter. After EOUT has been set to low (by SW or by a real fault), it loads (a value corresponding to T_min = 250 µs + value programmed in FCCU_DELTA_T register) and then starts down-counting and becomes 0 on the expiry of T_min window. It reloads on occurrence of another fault event which has EOUT signaling enabled. T_min feature is available only in Bi-Stable

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
   a) OP17
   b) OP19
   c) OP20
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the FCCU_XTMR register

Table 612: Timer state/value shows Timer state/value.

Offset: 0x0E8 Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XTMR_VAL															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	XTMR_VAL															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 877. FCCU XTMR Register (FCCU_XTMR)

Table 611. FCCU_XTMR field descriptions

Field	Description
31:0 XTMR_VAL	Alarm/Watchdog/Safe request timer The current timer value is measured in IRCOSC clock cycles. These bits can be read by the software.

Table 612. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	0x0000_0000	Initial value	Running	Idle/End of count
CFG	Running	0x0001_FFFF	0x0001_FFFF	0x0001_FFFF
EOUTMR	0x0000_0000	0x0000_0000	0x0000_0000	Running

51.6.22 FCCU transient register (FCCU_TRANS_LOCK)

The FCCU_TRANS register is used for unlocking configuration by writing 0xBC. This register is available only in supervisor mode. This register is write only.

Offset: 0x0F0 Access: Supervisor Write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0								
W									TRANSKEY							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 878. FCCU Transient Register (FCCU_TRANS_LOCK)

Table 613. FCCU_TRANS_LOCK field descriptions

Field	Description
7:0 TRANSKEY	Transition Unlocking Key value 0x00 No change ... 0xBB No change 0xBC Configuration unlocked 0xBD No change ... 0xFF No change

51.6.23 FCCU permanent lock register (FCCU_PERMNT_LOCK)

The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing 0xFF. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.

Offset: 0x0F4 Access: Supervisor Write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0								
W									PERMNTKEY							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 879. FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)

Table 614. FCCU_PERMNT_LOCK field descriptions

Field	Description
7:0 PERMNTKEY	Transition Locking Key value 0xFF Configuration Locked Any other value: No Change

51.6.24 FCCU delta T register (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds. This register can be written only in CONFIG state.

Offset: 0x0F8 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	Reserved													
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DELTA_T_VAL													
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 880. FCCU Delta T Register (FCCU_DELTA_T)

Table 615. FCCU_DELTA_T field descriptions

Field	Description
13:0 DELTA_T_VAL	Value of Delta_T (in microseconds) Max. value can be 10,000 µs (10 ms)

51.6.25 FCCU IRQ alarm enable register *n* (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm. This register is writable only in CONFIG state.

Offset: 0x0FC + *n**0x4 (*n* = 0 to 2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IRQEN31(1)	IRQEN30(1)	IRQEN29(1)	IRQEN28(1)	IRQEN27(1)	IRQEN26(1)	IRQEN25(1)	IRQEN24(1)	IRQEN23(1)	IRQEN22(1)	IRQEN21(1)	IRQEN20(1)	IRQEN19(1)	IRQEN18(1)	IRQEN17(1)	IRQEN16(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQEN15(1)	IRQEN14(1)	IRQEN13(1)	IRQEN12(1)	IRQEN11(1)	IRQEN10(1)	IRQEN9(1)	IRQEN8(1)	IRQEN7(1)	IRQEN6(1)	IRQEN5(1)	IRQEN4(1)	IRQEN3(1)	IRQEN2(1)	IRQEN1(1)	IRQEN0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 617* for register offset to channel number relationship.

Figure 881. FCCU IRQ Alarm Enable register *n* (FCCU_IRQ_ALARM_EN*n*)

Table 616. FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
31:0 IRQEN[31:0]	IRQ alarm enable 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

Table 617: FCCU IRQ alarm enable register channels shows FCCU IRQ alarm enable register channels.

Table 617. FCCU IRQ alarm enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x0FC	FCCU_IRQ_ALARM_EN0	IRQEN[31:0]
0x100	FCCU_IRQ_ALARM_EN1	IRQEN[63:32]
0x104	FCCU_IRQ_ALARM_EN2	IRQEN[95:64]

51.6.26 FCCU NMI enable registers *n* (FCCU_NMI_EN*n*)

These registers enable the NMI. These registers are writable only in CONFIG state.

Offset: 0x10C + *n**0x4 (*n* = 0 to 2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NMIEN31(1)	NMIEN30(1)	NMIEN29(1)	NMIEN28(1)	NMIEN27(1)	NMIEN26(1)	NMIEN25(1)	NMIEN24(1)	NMIEN23(1)	NMIEN22(1)	NMIEN21(1)	NMIEN20(1)	NMIEN19(1)	NMIEN18(1)	NMIEN17(1)	NMIEN16(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NMIEN15(1)	NMIEN14(1)	NMIEN13(1)	NMIEN12(1)	NMIEN11(1)	NMIEN10(1)	NMIEN9(1)	NMIEN8(1)	NMIEN7(1)	NMIEN6(1)	NMIEN5(1)	NMIEN4(1)	NMIEN3(1)	NMIEN2(1)	NMIEN1(1)	NMIEN0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 619* for register offset to channel number relationship.

Figure 882. FCCU NMI Enable register *n* (FCCU_NMI_EN*n*)

Table 618. FCCU_NMI_EN*n* field descriptions

Field	Description
31:0 NMIEN[31:0]	NMI enable 0 NMI is disabled for error (RF) source x. 1 NMI is enabled for error (RF) source x.

Table 619: FCCU NMI enable register channels shows FCCU NMI enable register channels.

Table 619. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x10C	FCCU_NMI_EN0	NMIEN[31:0]

Table 619. FCCU NMI enable register channels (continued)

Address offset	Register name	Channel range (x) (bit location [0:31])
0x110	FCCU_NMI_EN1	NMIEN[63:32]
0x114	FCCU_NMI_EN2	NMIEN[95:64]

51.6.27 FCCU EOUT signaling enable registers *n* (FCCU_EOUT_SIG_EN*n*)

These registers enable the error out signaling. Disabling of EOUT signaling is possible for Bi-stable protocol only. This register is writable only in CONFIG state.

Offset: 0x11C + n*0x4 (n = 0 to 2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EOUTEN31(1)	EOUTEN30(1)	EOUTEN29(1)	EOUTEN28(1)	EOUTEN27(1)	EOUTEN26(1)	EOUTEN25(1)	EOUTEN24(1)	EOUTEN23(1)	EOUTEN22(1)	EOUTEN21(1)	EOUTEN20(1)	EOUTEN19(1)	EOUTEN18(1)	EOUTEN17(1)	EOUTEN16(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EOUTEN15(1)	EOUTEN14(1)	EOUTEN13(1)	EOUTEN12(1)	EOUTEN11(1)	EOUTEN10(1)	EOUTEN9(1)	EOUTEN8(1)	EOUTEN7(1)	EOUTEN6(1)	EOUTEN5(1)	EOUTEN4(1)	EOUTEN3(1)	EOUTEN2(1)	EOUTEN1(1)	EOUTEN0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 621* for register offset to channel number relationship.

Figure 883. FCCU EOUT Signaling Enable register *n* (FCCU_EOUT_SIG_EN*n*)

Table 620. FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
31:0 EOUTEN[31:0]	EOUT signaling enable 0 EOUT signaling is disabled for error (RF) source x for bistable protocol. Error pins are as if in nonfaulty state.(1) 1 EOUT signaling is enabled for error (RF) source x.

1. In case of time switching and dual rail protocols, the error out signaling remains active even if the EOUT flag is 0b.

Table 621: FCCU EOUT signaling enable register channels shows FCCU EOUT signaling enable register channels.

Table 621. FCCU EOUT signaling enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x11C	FCCU_EOUT_SIG_EN0	EOUTEN[31:0]
0x120	FCCU_EOUT_SIG_EN1	EOUTEN[63:32]
0x124	FCCU_EOUT_SIG_EN2	EOUTEN[95:64]

51.6.28 FCCU NMI2 enable registers *n* (FCCU_NMI2_EN*n*)

These registers enable the NMI2. These registers are writable only in CONFIG state.

Offset: 0x12C + *n**0x4 (*n* = 0 to 2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NMI2EN31(1)	NMI2EN30(1)	NMI2EN29(1)	NMI2EN28(1)	NMI2EN27(1)	NMI2EN26(1)	NMI2EN25(1)	NMI2EN24(1)	NMI2EN23(1)	NMI2EN22(1)	NMI2EN21(1)	NMI2EN20(1)	NMI2EN19(1)	NMI2EN18(1)	NMI2EN17(1)	NMI2EN16(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NMI2EN15(1)	NMI2EN14(1)	NMI2EN13(1)	NMI2EN12(1)	NMI2EN11(1)	NMI2EN10(1)	NMI2EN9(1)	NMI2EN8(1)	NMI2EN7(1)	NMI2EN6(1)	NMI2EN5(1)	NMI2EN4(1)	NMI2EN3(1)	NMI2EN2(1)	NMI2EN1(1)	NMI2EN0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 623* for register offset to channel number relationship.

Figure 884. FCCU NMI2 Enable register *n* (FCCU_NMI2_EN*n*)

Table 622. FCCU_NMI2_EN*n* field descriptions

Field	Description
31:0 NMI2EN[31:0]	NMI2 enable 0 NMI2 is disabled for error (RF) source x. 1 NMI2 is enabled for error (RF) source x.

Table 623: FCCU NMI2 enable register channels shows FCCU NMI2 enable register channels.

Table 623. FCCU NMI2 enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x12C	FCCU_NMI2_EN0	NMI2EN[31:0]
0x130	FCCU_NMI2_EN1	NMI2EN[63:32]
0x134	FCCU_NMI2_EN2	NMI2EN[95:64]

51.7 FCCU output supervision unit (FOSU)

The FOSU provides a supervision of the primary error notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after an error arrives.

The intention of the FOSU is to provide a secondary error reaction path in most cases the FCCU fails but not to needlessly propagate an error which is already handled by the FCCU into a full chip reset. Only a failed primary error reaction (that is FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction. It is a "static" input in the sense that it does not change after FCCU configuration. This signal is automatically de-asserted whenever the FCCU moves to CONFIG state and hence FOSU must not be able to monitor faults occurring in CONFIG state.The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted (that is an enabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state. Also, the case of a continuously incoming disabled fault being enabled later, is not monitored.

The FOSU contains a timer with a duration of FOSU_COUNT,driven by the IRCOSC.The timer is initialized and started on any captured, enabled fault. While the timer is running, subsequent captured faults neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FCCU does not check whether the reaction is the configured one for the faults which occurred):

- Reset: Long or short functional reset.
- IRQ: NMI or Alarm.
- Error out triggered (by FCCU or by SW).

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still ‘stuck’ in the capture logic. The timer is then restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static “fccu program to do nothing” signal is de-asserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.
The FOSU's failure indicator output is connected to one of the RCC ‘destructive’ reset inputs, so its assertion causes a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as it is used by the FCCU (System reset), which is asserted on power-on, ‘destructive’, and long external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is de-asserted.

The following conditions show a use case of the FOSU:

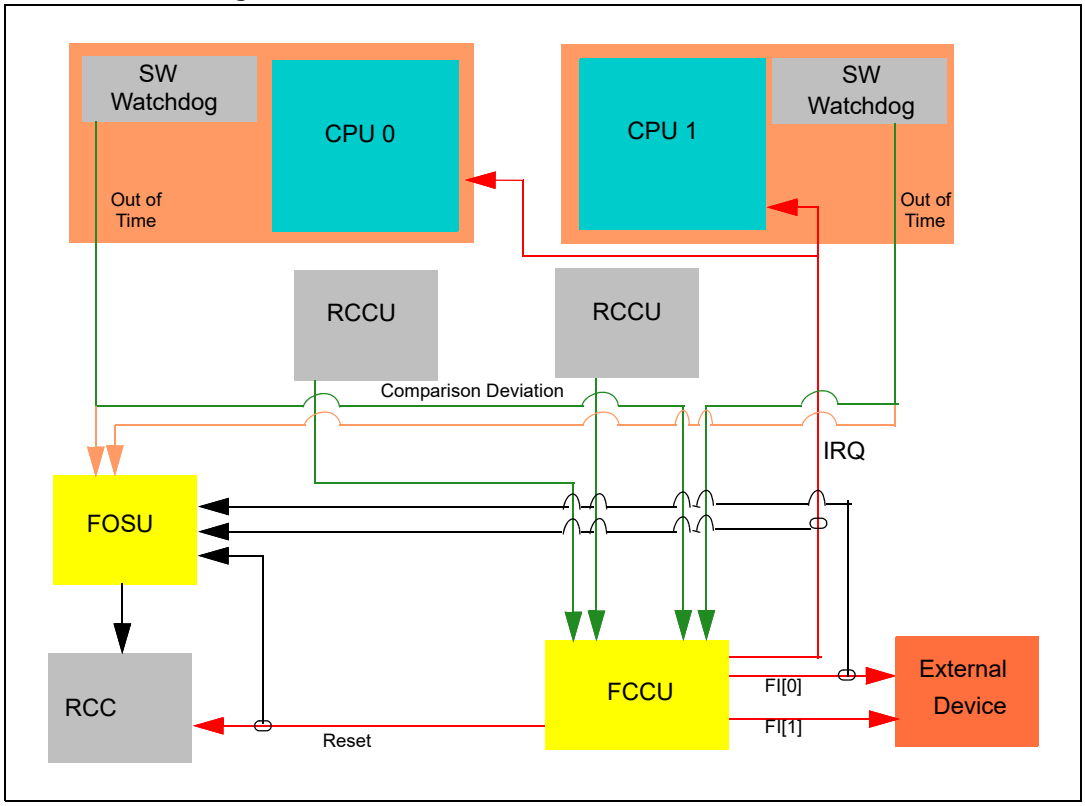
1. The user configures the channel x of the FCCU to react with a functional reset, either long or short.
2. The user sets the EOUTEN[x] flag of EOUT_SIG_EN register to 0.
3. The software does not clear the FCCU status flag related to the fault x after its first occurrence.
4. The fault x triggers again after the functional reset. Under these circumstances, the FOSU triggers a destructive reset after the FOSU timeout.

The rationale behind is that the functional reset is not able to clear the source of the fault. If the software can't react to the second occurrence of the fault before the FOSU timeout expires, the FOSU triggers a destructive reset.

If user wants to increase the availability of the system, when bi-stable protocol is selected, by avoiding destructive reset, it needs to program EOUT_EN[x] of EOUT_SIG_EN register to 1. In such a case, the FOSU does not trigger the destructive reset.

Note: FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times out without reaction. All intermediate faults are masked (not monitored) during this timeout cycle.

Figure 885. FOSU connections to the FCCU and RCC



It is important to remember that there is no FCCU reaction to an error while the FCCU is in the CONFIG state. For this reason, the FCCU must not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

51.8 Use cases and limitations

51.8.1 Misconfigurations

The following configurations are appropriate:

1. If at least one reaction to fault is enabled by default then the corresponding fault must also be enabled by default.
2. Enable a fault but disabling all reactions is not a meaningful configuration from a safety point of view.
3. Disable the fault and also its reactions. Applicable when a specific fault line is not of interest in a specific application scenario.
   Example: FCCU goes into ALARM state due to a failure, but the IRQ is disabled for that failure.
4. Software reset is optional.
5. The user must program the FCCU for a higher timeout value than the value hard-coded for FOSU. In this way FCCU reacts later than FOSU expects and as a result, FOSU generates a destructive reset request on its timeout, if a fault occurred when the FCCU was in CONFIG state.

51.8.2 Recommendations to configure FCCU

1. After Power on reset, where both system and FCCU get reset, the following steps could be followed to configure FCCU:
   a) Check and clear any pending fault status.
   b) Verify FCCU is in NORMAL state, else repeat step (a) above.
   c) Configure FCCU.
2. After functional reset of the system, arising out of reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a) If active, wait for the Error out Tmin to expire.
   b) Check and clear fault status.
   c) Error pin moves to “non faulty” state, once fault status is cleared and Tmin expires.
   d) Read and verify default value in FCCU_RF_Ex to ascertain reset of FCCU.
   e) Check and clear any new pending fault status until FCCU is in NORMAL state.
   f) Reconfigure FCCU.

52 Collective error manager (CEM)

52.1 Introduction

52.1.1 Overview

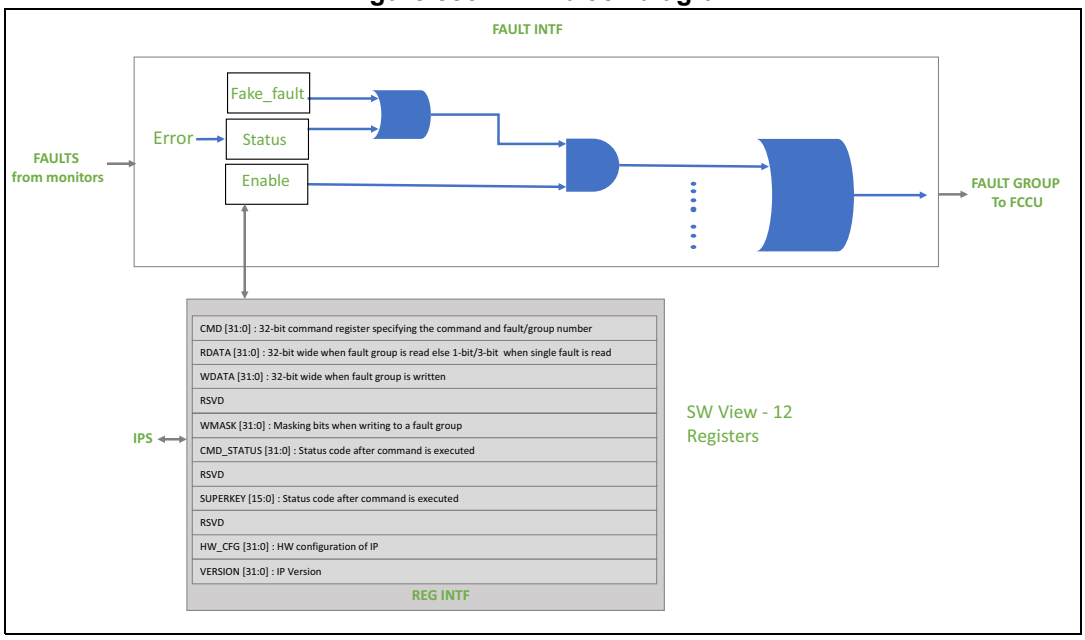
The collective error manager module contains registers dedicated to control and status reporting of errors from safety monitors to FCCU module. The errors are OR-ed together to generate one FCCU trigger. The module contains internal registers (per error group) for controlling and capturing status of errors from safety monitors, as well as fake fault injection, and IPS programmable registers for accessing internal registers.

IPS programmable registers (32-bit) are used to control and capture error status of the CEM internal registers.

52.1.2 Block diagram

Figure 886 shows SoC architecture for fault collection and injection via CEM module.

Figure 886. CEM block diagram



52.1.3 Features

- CEM functionality
  - CEM groups incoming faults (up to 32 faults per group) from safety monitors. The aggregated fault trigger is sent to FCCU
  - Reduced number of fault triggers to the FCCU
- CEM internal registers
  - CEM internal registers for status, fake fault injection and enable for control and status reporting of errors
  - Indirect access for CEM internal registers
  - False write protection enhanced due to indirect access of internal registers
  - Both bit-wise access and group access of CEM internal registers are supported
  - Status error bit is updated to indicate invalid internal register bit access
  - Trigger control logic to capture incoming asynchronous errors from safety monitors
- CEM programmable registers
  - IPS 32-bit programming interface, with 12-bit address and 32-bit data
  - Programmable write bit mask
  - Word access of programmable registers
  - Bus error on access to invalid programmable register
  - Up to 8 (32-bit) registers as control/status registers, which are read/write accessible (some status registers can be written in supervisor mode)
  - Active high status inputs
  - Key protected access, each access validated against “super” key
  - Version and HW configuration information
- CEM Clocking
  - Module operates on single clock (IPS clock)
- CEM reset
  - Destructive reset resets all programmable registers and internal CEM register bits
- Bus transfer error generation:
  - CEM supports key-protected access. Key field value (in case of CMD register access) of write data bus is validated against superkey value, and mismatch between the two values generates bus transfer error
  - For CMD register access, invalid command field value generates bus transfer error
  - Invalid register address selection (in case of programmable registers), bus transfer error is generated

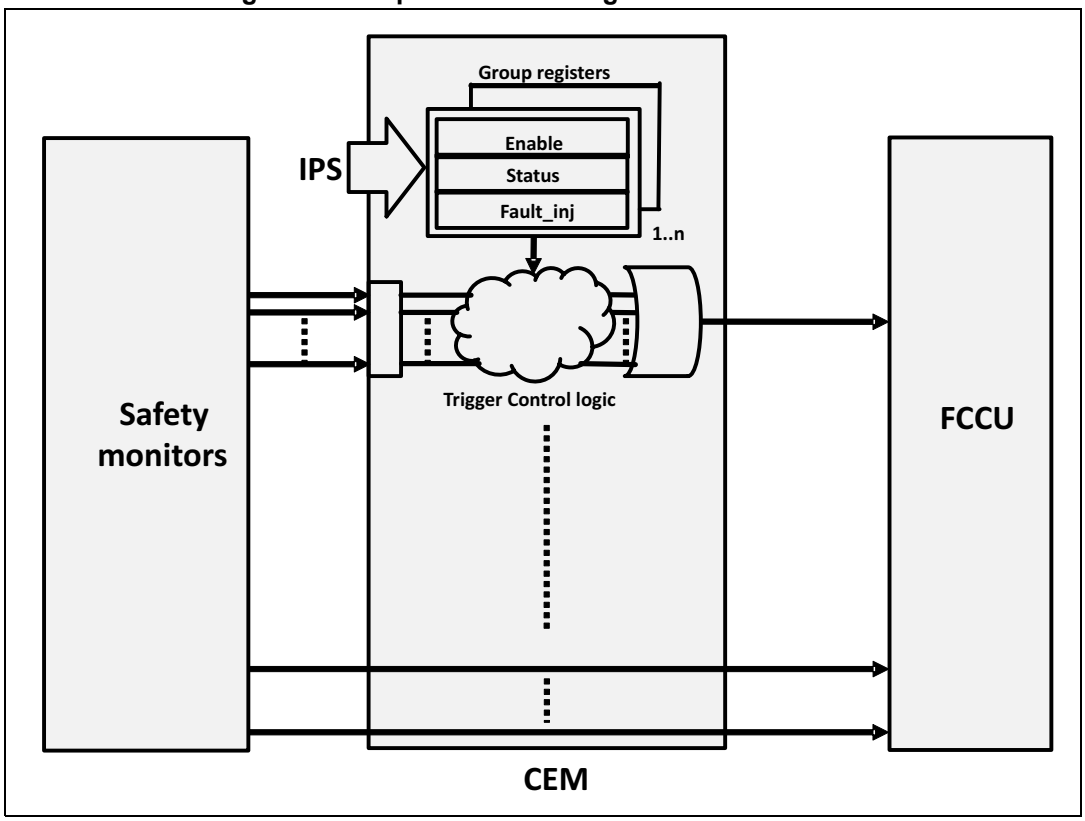
52.2 System interfaces

CEM module has IPS programmable interface for accessing programmable registers, which are in turn used for controlling and reporting status of CEM internal registers.

52.2.1 System block diagram

Figure 887 describes the top level system SoC architecture depicting implementation and overall architecture of CEM module for fault control and status capture of incoming faults from safety monitors to FCCU.

Figure 887. Top level block diagram of CEM module



52.2.2 Bus interface

IP bus interface is used for accessing programmable registers for control and status reporting. The internal registers are accessed indirectly via programmable registers.

Table 624. Bus interface

S.N.	Bus interface	Bus type	Description
1	IPS bus	Peripheral bus	32-bit address/data bus for programming CEM registers

52.2.3 IO interface

This IP does not interact with the external IO devices.

52.2.4 Interrupt interface

This IP does not have interrupts.

52.2.5 DMA interface

This IP does not have DMA requests.

52.2.6 Clock interface

This section gives information regarding the Clock interfaces of this IP.

Table 625. Clock interface

S.N	Clock name	Direction	Description
1	Peripheral clock	Input	IPS interface clock used for programming CEM registers

52.2.7 Reset interface

This section gives information regarding the Reset interfaces of this IP.

Table 626. Reset interface

S.N	Reset name	Direction	Description
1	Destructive/Peripheral reset	Input	System/Destructive reset for all registers (including internal registers)

52.2.8 Safety fault interface

This section gives information regarding the Fault interfaces of this IP.

Table 627. Safety fault interface

S.N	Fault name	Description
1	OUT_ERR [GRP_NUM-1:0]	Aggregated error signals from CEM to FCCU for groups 0 to GRP_NUM-1

52.3 Registers

52.3.1 Memory map

Table 628 shows the CEM memory map.

Table 628. CEM memory map

Offset (Hex)	Registers	Access	
		User	Supervisor
0x0000	*Command register (CMD)*	RW	RW
0x0004	*Read data register (RDATA)*	R	R
0x0008	*Write data register (WDATA)*	RW	RW
0x00010	*Write mask register (WMASK)*	RW	RW

Table 628. CEM memory map (continued)

Offset (Hex)	Registers	Access	
		User	Supervisor
0x0014	*Command status register (CMD_STATUS)*	R	R
0x0018 — 0x0023	**Reserved**		
0x0024	*Super key register (SUPERKEY)*	R(1)	RW
0x0028 — 0x002F	**Reserved**		
0x0030	*Hardware configuration register (HW_CFG)*	R	R
0x0034	*Version register (VERSION)*	R	R
0x0038 — 0x0040	**Reserved**		

1. User access for this register is read as ‘0’.

52.3.2 Register description

This section consists of CEM register descriptions.

52.3.2.1 Command register (CMD)

Command register is half-word accessible. In case, there is no change to key field, the remaining register can be accessed using byte-wise or half-word access.

Offset: 0x0000

Access: User read/write and Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	KEY															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD				FAULT_OR_GRP_NUM											
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 888. Command register (CMD)

Table 629. CMD field descriptions

Field	Description
31:16 KEY	Key for writing to CEM programmable registers, validated against the “super” key.
15:11 CMD	Command for accessing the CEM internal register (set/clear enable, set/disable fake fault injection, read status and so on)
10:0 FAULT_OR_GRP_NUM	Fault number or group number of corresponding internal CEM registers, for accessing individual error control/status info, as well as error group control/status info.

52.3.2.2 Read data register (RDATA)

Offset: 0x0004

Access: User read-only and Supervisor read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	READ_DATA															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	READ_DATA															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 889. Read data register (RDATA)

Table 630. RDATA field descriptions

Field	Description
31:0 READ_DATA	Data read after command is executed. If only single bit is accessed (fault_inj, status or enable), read_data[0] is the output. Bit 0: ENABLE Bit 1: FAULT_INJ Bit 2: STATUS

52.3.2.3 Write data register (WDATA)

Offset: 0x0008

Access: User read/write and Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WRITE_DATA															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WRITE_DATA															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 890. Write data register (WDATA)

Table 631. WDATA field descriptions

Field	Description
31:0 WRITE_DATA	Write data for setting/clearing internal RW register groups (fault_inj and enable). The mask bits may be applied to write to selected bits.

52.3.2.4 Write mask register (WMASK)

Offset: 0x0010

Access: User read/write and Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MASK															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MASK															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 891. Write mask register (WMASK)

Table 632. WMASK field descriptions

Field	Description
31:0 MASK	Mask bits for accessing individual fault group bits during write operations. 1 Masking of bit is enabled, that is, this bit is not updated. 0 Masking of bit is disabled.

52.3.2.5 Command status register (CMD_STATUS)

Offset: 0x0014

Access: User read-only and Supervisor read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	INV_CMD	INV_KEY	INV_REG
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 892. Command status register (CMD_STATUS)

Table 633. CMD_STATUS field descriptions

Field	Description
2 INV_CMD	Invalid command is written in the cmd.cmd field 0: Valid command is issued by the user 1: Invalid command is issued by the user
1 INV_KEY	Invalid key is written in cmd.key field 0: Valid key is used for access 1: Invalid key is used for access
0 INV_REG	Invalid address is being accessed 0: Valid address is provided for internal register access 1: Invalid address is provided for internal register access

52.3.2.6 Super key register (SUPERKEY)

Offset: 0x0024

Access: User read-only(1) and Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SUPERKEY															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 893. Super key register (SUPERKEY)

1. User access is read as ‘0’.

Table 634. SUPERKEY field descriptions

Field	Description
15:0 SUPERKEY	Golden key used for validation during write access. Write access only during supervisor mode.

52.3.2.7 Hardware configuration register (HW_CFG)

Offset: 0x0030

Access: User read-only and Supervisor read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HW_CFG_PARAM															
Reset:	Refer to device configuration chapter for specific bit reset value.															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HW_CFG_PARAM															
Reset:	Refer to device configuration chapter for specific bit reset value.															

Figure 894. Hardware configuration register (HW_CFG)

Table 635. HW_CFG field descriptions

Field	Description
31:0 HW_CFG_PARAM	Hardware configuration parameter, for each IP release. This information is provided for debug purposes. The value of this register is dependent on number of faults.

Note: Reset value of HW_CFG register is device dependent. Refer to the device configuration chapter.

52.3.2.8 Version register (VERSION)

Offset: 0x0034

Access: User read-only and Supervisor read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	VERSION															
Reset:	Refer to device configuration chapter for specific bit reset value.															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VERSION															
Reset:	Refer to device configuration chapter for specific bit reset value.															

Figure 895. Version register (VERSION)

Table 636. VERSION field descriptions

Field	Description
31:0 VERSION	Release version of IP. This information is provided for debug purposes. The value is fixed per CEM instance, depending on the release number.

Note: Reset value of VERSION register is device dependent. Refer to the device configuration chapter.

52.4 Functional description

52.4.1 CEM functionality

The CEM acts as a fault aggregation layer. The CEM module groups the incoming faults (up to 32 faults per group) from safety monitors, thus reducing the number of fault triggers to FCCU (due to limited number of FCCU input pins).

CEM also provides a mechanism to mask the incoming faults from safety monitors to FCCU, by selectively enabling/disabling each error channel.

Also CEM generates fault triggers with uniform characteristics, independently of the characteristics of each safety monitor.

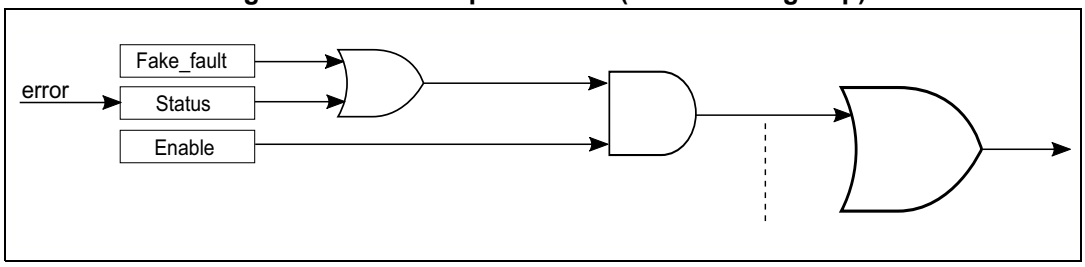
52.4.2 CEM internal registers

The CEM contains internal registers dedicated to control and report status of error from safety monitors to FCCU.

52.4.2.1 Fault capture logic

The incoming errors from the safety monitors are organized into groups (maximum 32 errors per group). The number of faults are parameterized, as per the SoC implementation. The control and status report of incoming error within group is depicted in *Figure 896*. The status bit captures the real error coming from the safety monitors. The fault is captured asynchronously.

Figure 896. Fault capture block (within error group)



The existence of error within group, and existence of fake fault mechanism inside CEM module are parameterized as well.

The CEM internal register bits (per fault channel) are described as below:

1. ENABLE: Enable/disable the propagation of incoming fault from safety monitor to FCCU. The enable bit is set upon reset (default).
2. STATUS: The incoming fault from safety monitor is latched on to status register bit.
3. FAULT_INJ: Fake fault injection via SW into FCCU.

The CEM internal registers are indirectly accessed via programmable registers. This provides immunity to false-write to some extent.

52.4.3 CEM Programmable register

The CEM module contains control/status registers for accessing internal registers of the module. The CEM registers are IPS (12-bit address/32-bit data) programmable. Registers are byte, half-word or word accessible, any other access combination yields error.

The CEM command register field FAULT_OR_GRP_NUM is used to identify the individual fault/fault group to be accessed. This provides access to individual errors as well as complete error group. The CMD field is decoded to identify the type of operation to be performed on the internal registers. Key protection is provided for write access to the internal registers to enhance the safety of the module. The KEY field is validated against the SUPERKEY register value for each write access. The SUPERKEY register is configurable in the supervisor mode (default value is 0xA5A5).

In case a selective number of errors needs to be accessed at given time, write bit mask is used to selectively mask some bits from getting accessed within a group. The status of command execution is captured in the CMD_STATUS register. All the status signals are active high.

The module also provides IP release version information, and parameter value used for configuration as debug feature, which can be later used for debugging across multiple devices, which used different configuration/release version. No bus transfer error is generated upon access to read-only registers for write purposes, or while accessing SUPERKEY register in user mode.

52.4.4 Clock and reset

The CEM module operates on single IPS clock.

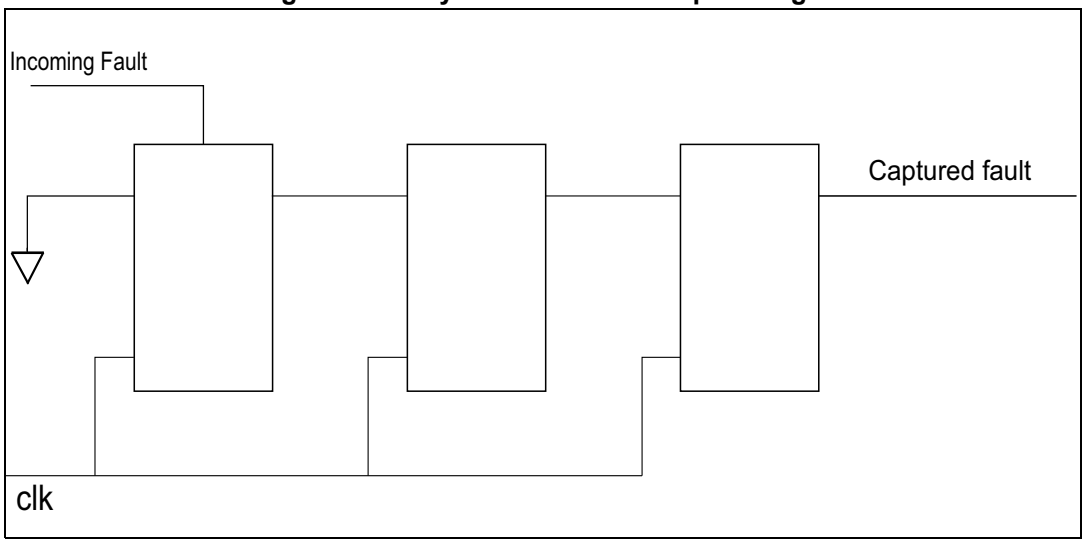
The destructive reset is used to reset the entire CEM module (including programmable interface and the CEM fault control and status logic), while all the enable bits are set.

Upon destructive reset the command status register initially indicates invalid key and command, as the default values of command field and key field are not accepted.

52.4.5 Asynchronous fault capture

In order to asynchronously capture the incoming fault signal from the safety monitors to the CEM module, an asynchronous fault capture scheme is implemented.

Figure 897. Asynchronous fault capture logic



52.4.6 Debug mode behavior

As the internal registers are indirectly accessed in the CEM module, in the debug mode, a sequence of commands needs to be executed, to view the content of the internal registers.

52.5 Safety mechanism

Safety mechanism/features:

- Key protection: Each write access is done with safety key, which is validated against the super key/golden key.
- Indirect access to internal registers: This minimizes the chances of false writes to internal register bits.

52.6 Security

Detailed information on the IP security architecture is published in a separate document and is available on a limited basis to customers who have a non-disclosure agreement (NDA) with STMicroelectronics.

52.7 Programming guidelines

Commands ENABLE_FAULT, DISABLE_FAULT, GROUP_READ_ENABLE and GROUP_WRITE_ENABLE make it possible to read and write the channel enabling status. Writing must be done in privilege mode only.

Commands READ_STATUS, CLEAR_STATUS, WRITE_STATUS, GRP_READ_STATUS, GRP_W1C_STATUS and GRP_W1S_STATUS make it possible to read and write the channel fault status. Writing must be done in privilege mode and it must be write-1-clear.

Commands SET_INJECT, CLEAR_INJECT, GROUP_READ_INJECT and GROUP_WRITE_INJECT make it possible to read and write the channel fault injection. Writing must be done in privilege mode only.

Table 637. CEM commands

Command	Label	Description
0x1	ENABLE_FAULT	Enable a specific input fault channel.
0x2	DISABLE_FAULT	Disable a specific input fault channel.
0x3	SET_INJECT	Inject a fault into a specific input fault channel.
0x4	CLEAR_INJECT	Clear the fault injected into a specific input fault channel.
0x5	READ_STATUS	Read the fault status of a specific input fault channel. The status is returned in RDATA[0].
0x6	CLEAR_STATUS	Clear the fault for a specific input fault channel. **Note:** Always run READ_STATUS command after running CLEAR_STATUS to ensure that the status is successfully cleared. This also tells the CEM to enable back any future errors on this input channel and not clear them.
0x7	READ_FAULT_DATA	Read the ENABLE, INJECT, and STATUS bits for a fault channel. Data is reported in the RDATA on bits 0 (ENABLE), 1 (INJECT), and 2 (STATUS).
0x10	GROUP_READ_ENABLE	Read the enabling status of the input fault channels for a specific OR-group The read data is reported in the RDATA register.
0x11	GROUP_WRITE_ENABLE	Enable or disable the input fault channels for a specific OR-group. WDATA is used to set which CEM input channel must be enabled.
0x12	GROUP_READ_INJECT	Read the fault injection status of the input fault channels for a specific OR-group The read data is reported in the RDATA register.
0x13	GROUP_WRITE_INJECT	Set or clear the injection faults for the input fault channels for a specific OR-group WDATA is used to set on which CEM input channel a fault must be injected.
0x14	GROUP_READ_STATUS	Read the fault status of the input fault channels for a specific OR-group The read data is reported in the RDATA register.
0x15	GROUP_W1C_STATUS	Write (write-1-clear) the fault status of the input fault channels for a specific OR-group. WDATA is used to set which CEM input channel status bits must be write-1-cleared. **Note:** Always run GROUP_READ_STATUS command after running GROUP_W1C_STATUS to ensure that the status is successfully cleared. This also tells the CEM to enable back any future errors on these selected input channels and not clear them.
0x16	GROUP_W1S_STATUS	Write (write-1-set) the fault status of the input fault channels for a specific OR-group. WDATA is used to set which CEM input channel status bits must be write-1-set.

Table 638. Programming guidelines

Command opcode	Command name	Read/ Write	Bit/ Group	Sequence
0x1	ENABLE_FAULT	Write	Bit	1. Write COMMAND register w/ Opcode + Fault number + Key. 2. Read COMMAND register. **Note:** In case of running CLEAR_STATUS, also run READ_STATUS command after running CLEAR_STATUS.
0x2	DISABLE_FAULT			
0x3	SET_INJECT			
0x4	CLEAR_INJECT			
0x6	CLEAR_STATUS			
0x5	READ_STATUS	Read	Bit	1. Write COMMAND register w/ Opcode + Fault number + Key. 2. Read RDATA register to get the required value.
0x7	READ_FAULT_DATA			
0x11	GROUP_WRITE_ ENABLE	Write	Group	1. Write WDATA + WMASK. 2. Write COMMAND register w/ Opcode + Group number + Key. 3. Read COMMAND register. 4. Check the output op_err. 5. Read RDATA register.
0x13	GROUP_WRITE_ INJECT			
0x15	GROUP_W1C_ STATUS	Write	Group	1. Write WDATA (for example: 0xFFFF_FFFF to set/ clear all bits of group). 2. Write COMMAND register w/ Opcode + Group number + Key. 3. Read COMMAND register. 4. Check the output op_err. 5. Read RDATA register. **Note:** In case of running GROUP_W1C_STATUS, also run GROUP_READ_STATUS command after running GROUP_W1C_STATUS.
0x16	GROUP_W1S_ STATUS			
0x10	GROUP_READ_ ENABLE	Read	Group	1. Write COMMAND register w/ Opcode + Group number + Key. 2. Read RDATA register to get the required value.
0x12	GROUP_READ_ INJECT	Read	Group	1. Write COMMAND register w/ Opcode + Group number + Key. 2. Read RDATA register to get the required value.
0x14	GROUP_READ_ STATUS	Read	Group	1. Write COMMAND register w/ Opcode + Group number + Key. 2. Read RDATA register to get the required value.

Examples of a CEM internal register bit/group access via IPS programmable registers:

1. Enable the CEM input Fault 129: Write CMD Register = 0xA5A5_0881
2. Disable the CEM input Fault 129: Write CMD Register = 0xA5A5_1081
3. Read Status of the CEM input Fault 129:
   Write CMD Register = 0xA5A5_2881
   Read BIT-0 of RDATA Register
4. Read all enabled input faults in CEM Group32:
   Write CMD Register = 0xA5A5_8020
   Read RDATA Register
5. Read Fault Data of the CEM input fault 129:
   Write CMD Register = 0xA5A5_3881
   Read RDATA Register: Bit0=Enabling, Bit1=Injection, BIT2=Input Status

53 Memory error management unit 2 (MEMU2)

53.1 Introduction

The MEMU2 is responsible for collecting and reporting error events to the Fault Collection and Control Unit (FCCU) associated with faults detected by memory BISTs as well as ECC (Error Correction Code) logic, used on:

- System-accessible RAM
- Peripheral local RAM
- Non-Volatile Memory (NVM)

The MEMU2 records and reports the following categories of errors:

- Correctable error:
  - Single bit errors detected for System RAM, Peripheral local RAM, or NVM access.
  - Double bit errors detected for NVM.
- Uncorrectable error:
  - Multiple bit errors for a system RAM, peripheral local RAM, or more than two-bit errors for NVM.

The MEMU2 system connections are chip-specific; see the Device configuration chapter that describes how modules are configured and connected.

MEMU2 processes the errors based on whether they are correctable or uncorrectable (from either ECC or MBIST logic), independent of the source generating these errors.

The MEMU2 has multiple instances, one each for:

- System RAM ECC and MBIST
- Peripheral SRAM ECC
- NVM ECC.

Each instance has two reporting tables: correctable errors are reported in one table while the uncorrectable are reported in the other. NVM reporting tables are further different for single-bit correctable and double-bit correctable errors.

See *Section 53.4: Design overview* for the reporting block details.

53.2 Features

The MEMU2 has the following features:

- Support for error reporting from the following category of error sources (both for ECC and MBIST):
  - System RAM
  - Peripheral SRAM
  - NVM
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped register interface.
- MEMU2 handles overflow during error assertion and reports status accordingly (refer to *Section 53.7.3: Handling overflows*).
- Errors are stored in correctable and non correctable section of the reporting table (only if the error was not already logged) and corresponding indication is generated to the FCCU.
- The MEMU2 table can also be updated via the register interface to let the software add or remove error entries.
- Region filtering for ECC errors, by which error flags to FCCU can be routed to different FCCU channels depending on the type of reaction the error requires. The type of reaction depends on the memory source of that error.

53.3 Block diagram

Figure 898 shows the block diagram of a single instance of MEMU2.

Figure 898. MEMU2 single instance block diagram

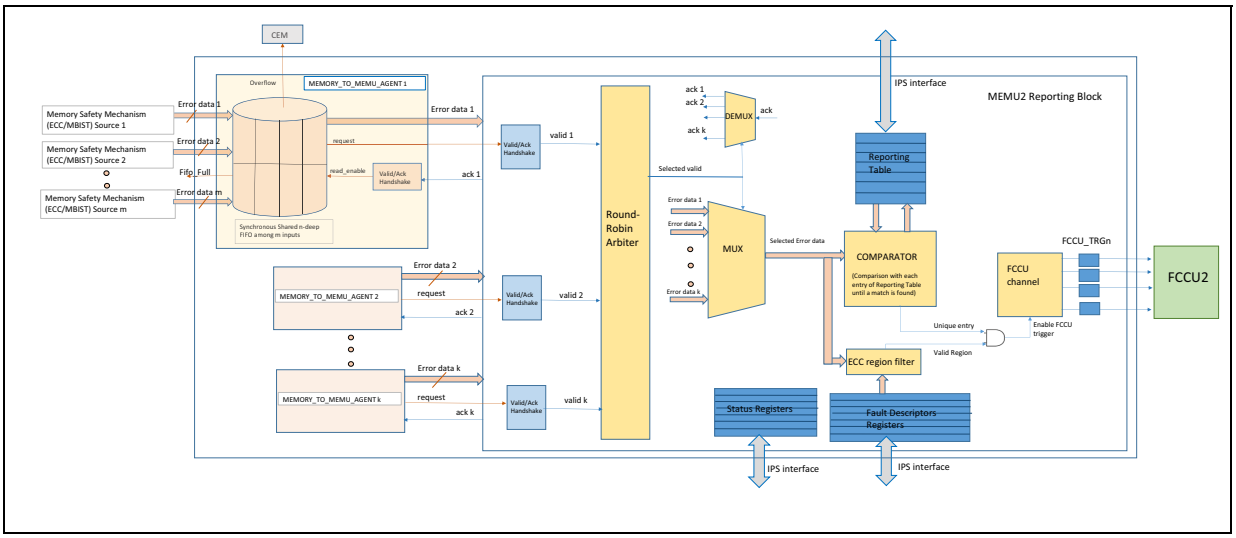


Figure 899 shows the system-level diagram of MEMU2, where the MEMU2 block has three instantiations along with some shared registers among the 3 instances. A common CPU interface is present for the user software to access the common registers as well as registers inside each MEMU2 reporting block.

Figure 899. System-level diagram of MEMU2

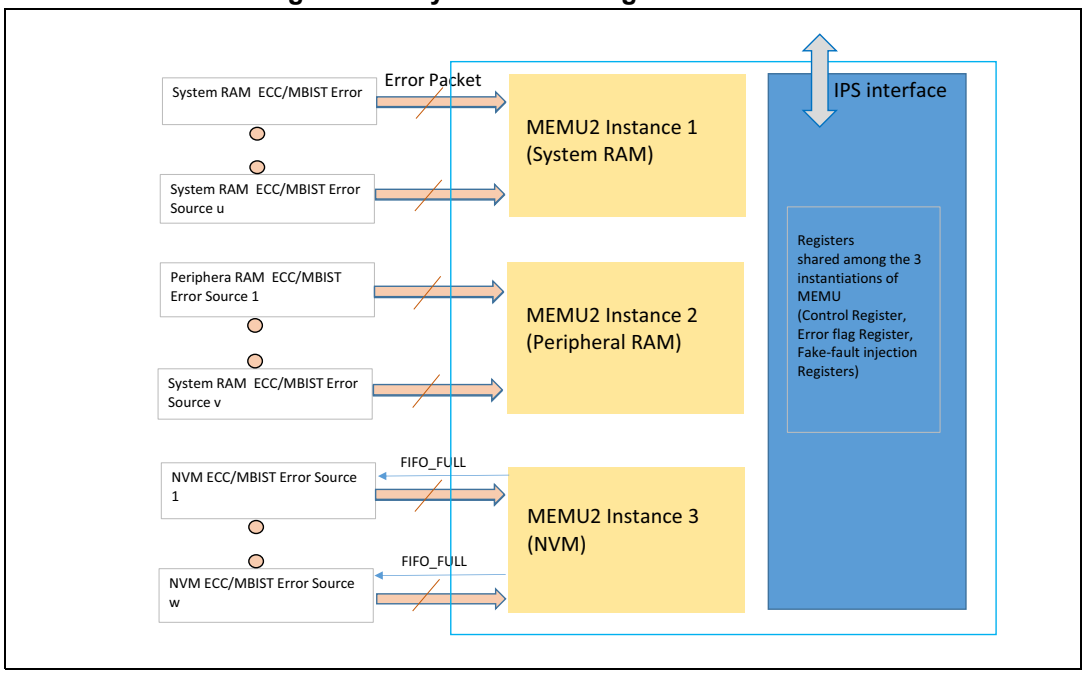
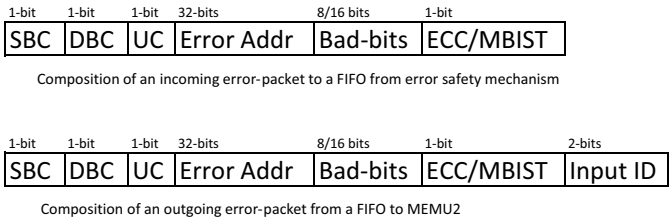


Figure 900 depicts the contents of error-packet incoming to a FIFO and outgoing from a FIFO to MEMU2.

Figure 900. Error packet



53.4 Design overview

The basic architecture for error reporting in the MEMU2 is summarized in the following sections.

53.4.1 Memory to MEMU agent

A synchronous FIFO-based structure is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits/syndrome) are concatenated and stored in the FIFO. The FIFO is synchronous by nature, and both write and read to or from the FIFO works on the same clock as the safety mechanism of the corresponding memory. The FIFOs are placed as close as possible to their respective memory safety mechanism module.

The FIF0 is n-deep and is shared among m possible safety monitors. For example, 8-deep FIFO shared among 4 input error sources. The depth of FIFO is parametrized. The maximum number of input sources that can be handled by a single FIFO is 4.

Whenever there is an overflow in the synchronous FIFO, it is denoted by sending a FIFO_overflow flag to FCCU via CEM.

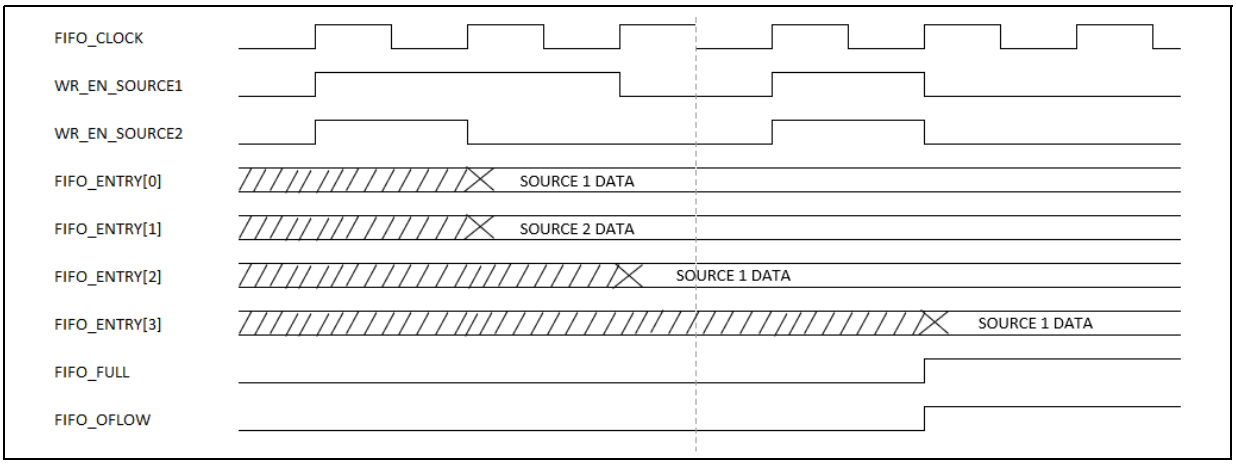
The FIFOs allocated for NVM Safety Mechanism also sends a back-pressure (FIFO_FULL) signal to the NVM safety mechanism module.

A handshake mechanism is used to communicate errors to MEMU2 from the FIFO. When a FIFO has data, it sends a request to MEMU2, indicating that it has a valid error. If the error sequencer inside MEMU2 selects this error for processing, an acknowledge is sent to the FIFO, whose error was accepted by MEMU2. On reception of the acknowledge, the read pointer of the FIFO points to the next valid data.

The above mentioned FIFO and handshake scheme is used by both asynchronous and synchronous memory-error sources. However, in case of asynchronous sources, proper synchronization is done for request and acknowledge signals and error data crossing clock domains.

Figure 901 shows the FIFO write behavior.

Figure 901. FIFO write



53.4.2 Error processing block

This block is used to process the incoming errors coming from the device. The error sequencer is used for sequencing the errors so that one error is processed per clock cycle. A simple round-robin arbitration scheme is used to select the error which is sent for processing. If say, N error sources assert an error valid request to MEMU2 in the same cycle, then the arbiter chooses the error corresponding to the lowest index number. Then, in the next cycle, the arbitration process begins from the previously chosen index+1.

Up to 64 error channels per MEMU2 reporting block (System RAM/Peripheral RAM/NVM) are supported. The number of error channels are chip-specific; see the Device configuration chapter that describes how modules are configured and connected.

An acknowledge information is sent to the Memory_to_MEMU agent, whose error was selected for processing.

53.4.3 Reporting table

This is a predefined table in the MEMU2 to store the details for the device.

The type of error is determined and stored in the correct reporting table (correctable or uncorrectable).

Each entry of the reporting table is associated with a valid flag, which is set once an error is latched and stored into the table.

For the reporting table, if the entry is unique, it is stored in the reporting table, else it is discarded. For this search, only the entries in the reporting table are considered, which have the VALID flag set to 1 and are from the same error source (ECC or MBIST) as the incoming error. The fault comparison is based on the following:

a) Fault address(32 bits)
b) Syndrome (16 bits for NVM, 8 bits otherwise)
c) Fault type (1 bit to signal a correctable error, 1 bit to signal an uncorrectable error. NVM errors provide 2 separate bits for single and double correctable errors).

In case of uncorrectable errors, there is no syndrome/bad-bit information present, neither are the errors categorized into ECC/MBIST when stored in the reporting table. Thus, for the uniqueness check of uncorrectable error, only address comparison is done. If the address of the incoming error is the same as the address of a valid error in the uncorrectable-reporting table, the error is discarded.

If a table has been completely filled and a new error occurs, then a fault trigger to the FCCU is activated, indicating table overflow. The fault trigger is separated for each table.

The table entries of the reporting table are exposed as separated registers with a valid bit. It is possible for the software to clear the valid bit, thus making the table entry invalid, and thus it can be reused.

The user software can write known error addresses into the reporting table to prevent from reporting of those errors addresses to FCCU must they later be accessed in operation.

The number of valid entries are counted for correctable error reporting tables for each NVM, System RAM and peripheral RAM. Dedicated registers store this information.

The CPU can clear the flags signaling errors or reporting table overflows to the FCCU directly by writing 1 at the corresponding bit of the ERR_FLAG register.

The clearing of flags is totally independent from the status of VLD bits in the reporting table.

The reporting table sizes are chip-specific; see the Device configuration chapter that describes how modules are configured and connected.

53.4.4 Register block

The CPU register programming interface provides the memory-mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. However, in the situation when both MEMU2 and the CPU want to write to the same register location/same reporting table at the same time, MEMU2 must be given priority. If MEMU2 hardware wants to write to a particular reporting table, then a software write access to any of the entries of that reporting table in the same cycle is dishonored.In a situation where CPU wants to read a register at the same time when MEMU2 is writing to it, a wait cycle is signaled to the CPU and the CPU command is completed in the next clock cycle.

There is a feature of fake-fault injection, by which user software can force the various flags to FCCU to 1. This can help in checking connectivity between MEMU2 and FCCU.

The software, in principle, is not supposed to write VLD to 1 if it is already 1. If it spuriously does, then the corresponding field of error flag register is set to 1, in case the software had cleared the flag bit.

The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

53.4.5 ECC filtered reaction

53.4.5.1 Rationale

The ECC filtering scheme is dictated by the need to partition the application memory with different safety levels, that is from full ASIL-D down to QM.

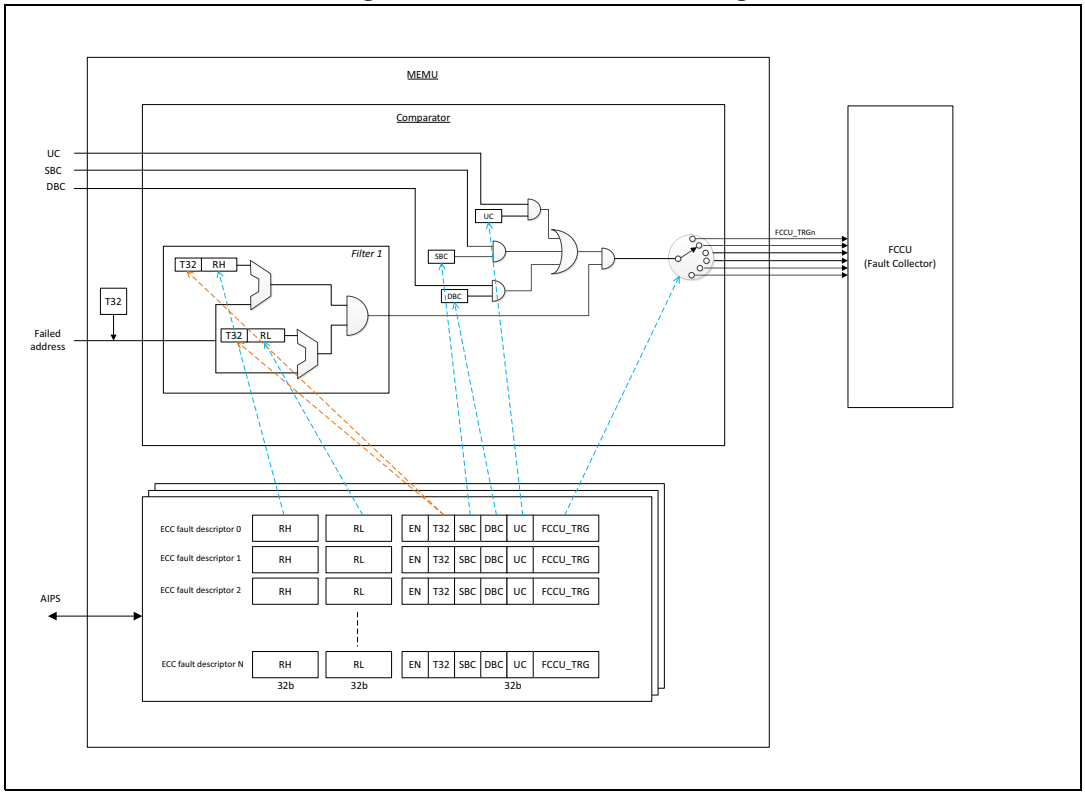
This requires that, upon an ECC failure, the reaction is treated in a different way, depending on:

- where the failure happened, that is which memory region
- the type of failure, that is single, double, or uncorrectable bit errors

53.4.5.2 Architecture

The following figure shows the related architecture.

Figure 902. MEMU2 ECC filtering



There are 3 instances of the above structure, one for NVM ECC faults, one for System ECC faults, and one for Peripheral Private RAM ECC faults.

Hereafter the description is intended to apply to one instance, but obviously it is also valid for the other instances.

For the sake of simplicity, the figure shows one single ECC fault.

In order to save time, the ECC filtering is executed in parallel with the logic that analyzes whether the fault already exists or is new.

The MEMU2 output signals for the Error Collector module is triggered only if:

1. The fault is new.
2. A valid region is found.

Note: If the software directly makes a valid entry in the MEMU2 reporting table, such errors do not generate any FCCU trigger due to region filtering.

It is possible that the error-table is full and the error could not get logged in the MEMU table, despite being new and in a valid region. In this case, 2 types of trigger are generated and sent to FCCU: overflow trigger as well as region-filtered trigger.

The address reported to the MEMU2, when used for the ECC filtering, is extended with a 32^{nd} bit, that is the compared address is 33-bit long.
This bit is called T32 (which stands for “Tag-32”) and it is a bit configurable via register interface (see *Section 53.6: Memory map and registers definition*) and allocated for each separated MEMU2 input fault channel.

The design uses parallel comparators. Every comparator is configured with a fault descriptor. The comparator verifies whether the failed address falls into a certain region and the fault type (correctable single bit, correctable double bit, uncorrectable multi bits) corresponds to the one enabled in the ECC fault descriptor. An incoming fault is compared with all the descriptors in parallel, and is routed to the corresponding FCCU channel if a match is found.

If MEMU2 receives an error, before the software programs the fault-descriptor registers, it is routed to the 0th FCCU channel (for each of System RAM, Peripheral RAM and NVM). These errors could be due to off-line MBIST or ECC errors from NVM. The reset value of the 0th fault descriptor for each of the three categories of memory allows such errors to trigger FCCU channel 0.

Each ECC fault descriptor holds three 32-bit registers:

- Start address
- End address
- Control

The Control register contains several fields:

1. EN: used to enable the ECC fault descriptor. If not enabled, it is simply ignored during the sequential access to the various ECC fault descriptors
2. T32: a 33^{rd} address bit, which is appended as a MSB bit to the 32 address reported to the MEMU2. The usage of this bit is to make it possible to minimize the number of ECC fault descriptors needed for memories mapped into non-consecutive addresses.

   For example, memory A and memory B are two non-consecutive physically separated memories that report ECC faults to separated MEMU2 input channels within the same fault class. As an example, memory A could be the DMA private RAM and memory B could be the GTM private RAM. In this example ECC faults on memories A and B must

be treated in the same way.
Memory C is another physically separated memory that reports ECC fault to a dedicated MEMU2 input channel. In this example, memory C is mapped between memories A and B and requires a different ECC fault treatment. As an example, memory C could be FlexRay private RAM. For each of these memories it is possible to set the T32 bit, which is appended to the reported faulty memory address, and it is possible to create a 33-bit “virtual” address for memories A, B, and C, where memory C is not anymore “seen” by the MEMU2 ECC filtering logic between memories A and B. By doing this, it is possible to combine the homogeneous ECC fault treatment of memories A and B using one single ECC fault descriptor.

3. SBC: Single-Bit correctable fault. It determines whether Single-Bit correctable ECC errors must be considered (if set to 1) or ignored (if set to 0).
4. DBC: Double-Bit correctable fault. It determines whether Double-Bit correctable ECC errors must be considered (if set to 1) or ignored (if set to 0). This bit is instantiated only for MEMU2 ECC input fault channels connected to the NVM.
5. UC: Uncorrectable fault. It determines whether uncorrectable ECC errors must be considered (if set to 1) or ignored (if set to 0).
6. FCCU_TRG: It determines which FCCU trigger number is activated. The programmed value corresponds to the output trigger number driven by the MEMU2 to the FCCU. If this field is set to “all-1”, no FCCU trigger is generated. This case is needed to be able to latch ECC errors in some memory regions, but where no specific treatment is required.

53.4.5.3 T32 usage example

Figure 903 shows an example to clarify the usage of the T32 bit flag.

The example shows 5 peripheral private memories (A, B, C, D, E) which are mapped to a certain memory space:

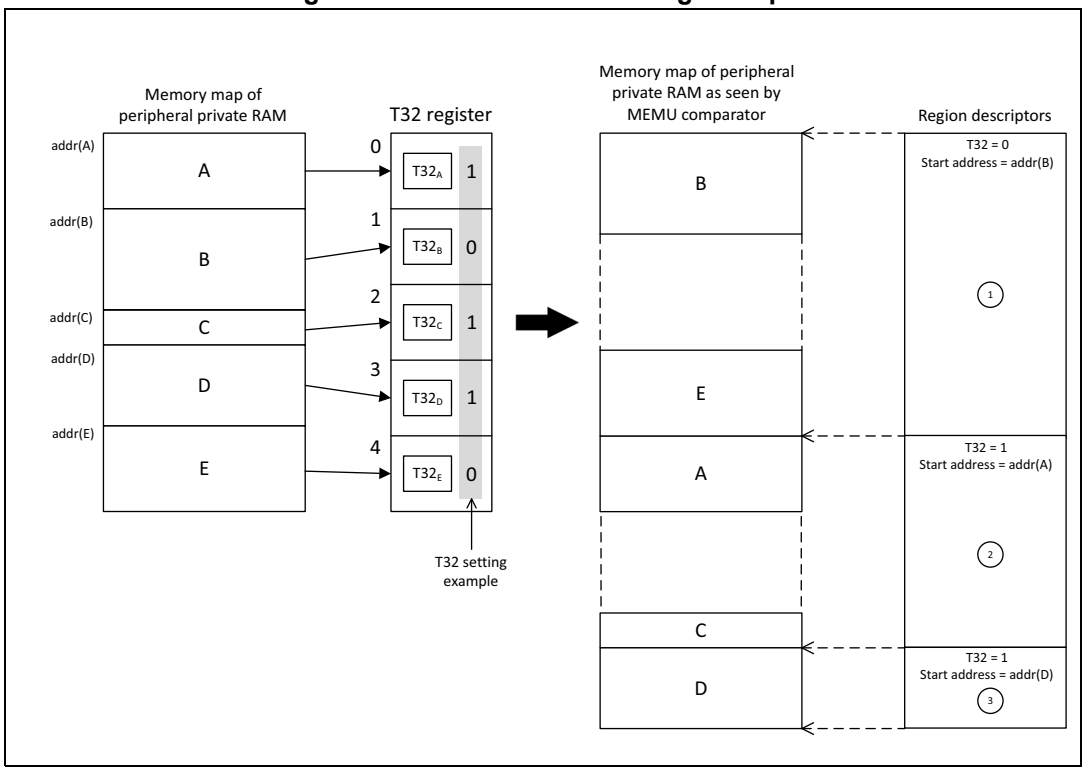
- Faults on memories B and E are treated in a certain way, that is they end up to a specific FCCU trigger.
- Faults on memories A and C are treated in another way, that is they end up to a different FCCU trigger.
- Faults on memory D have again a different treatment, so an additional FCCU channel is used.

Without using the T32 bit feature, the above scenario would need 5 different descriptors.

Instead, by properly assigning the T32 bit of each memory token, it is possible to put memories B and E as part of a first region descriptor, memories A and C as part of a second region descriptor, and finally memory D pointed by the third region descriptor.

In total three region descriptors are used in place of five.

Figure 903. MEMU2 ECC filtering example



53.5 External signal description

MEMU2 has no external signals.

53.6 Memory map and registers definition

53.6.1 Memory map

The memory map comprises 32-bit aligned registers and read-only registers that can be accessed via 8-bit,16-bit, or 32-bit accesses. Write access to reserved address locations generates a transfer error. Read data from reserved locations also generates a transfer error and the read data bus returns all 0's.

The Memory Error Management Unit (MEMU2) allows error reporting tables to be written by the CPU to simulate a memory error condition or to disable memory errors from a known faulty location. This requires that the valid bit for the System RAM, Peripheral RAM, or NVM in the error reporting table (SYS_RAM_UNCERR_STS[VLD], PERIPH_RAM_UNCERR_STS[VLD], NVM_UNCERR_STS[VLD]) corresponding to the address register (SYS_RAM_UNCERR_ADDR, PERIPH_RAM_UNCERR_ADDR, NVM_UNCERR_ADDR) is asserted (set to 1) when the address is written to the register.

Note: The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

The Control, Error Flag and Output Trigger Control registers are common for all MEMU2 error reporting blocks. The correctable error status, correctable error address, uncorrectable

error status, uncorrectable error address and concurrent error overflow registers are replicated for each MEMU2 instance.

Note: The actual availability, number of registers and number of reported errors in the reporting table are chip-specific; see the Device configuration chapter that describes how modules are configured and connected.

Table 639. MEMU2 memory map

Offset or address	Register	Access	Reset value	Page
0x0000	Control register (CTRL)	R/W	0x0000_0000	*Section 53.6.2.1*
0x0004	Error Flag register (ERR_FLAG)	R/W	0x0000_0000	*Section 53.6.2.2*
0x000C	System RAM output trigger control register (SYS_RAM_OUT_TRIG_CTRL)	R/W	0x0000_0000	*Section 53.6.2.3*
0x0010	Peripheral RAM output trigger control register (PERIPH_RAM_OUT_TRIG_CTRL)	R/W	0x0000_0000	*Section 53.6.2.4*
0x0014	NVM output trigger control register (NVM_OUT_TRIG_CTRL)	R/W	0x0000_0000	*Section 53.6.2.5*
0x0018-0x001C	Reserved			
0x0020 + *n**0x8 (*n* = 0 to 10 - 1)	System RAM correctable error reporting table status registers (SYS_RAM_CERR_STS*n*)	R/W	0x0000_0000	*Section 53.6.2.6*
0x0024 + *n**0x8 (*n* = 0 to 10 - 1)	System RAM correctable error reporting table address registers (SYS_RAM_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 53.6.2.7*
0x0028 + (10 - 1)*0x8	System RAM correctable error reporting table fill status register (SYS_RAM_CERR_TBL_FILL_STAT)	RO	0x0000_0000	*Section 53.6.2.8*
0x002C + (10 - 1)*0x8 - 0x042C	Reserved			
0x0430 + *n**0x8 (*n* = 0 to 1 - 1)	System RAM uncorrectable error reporting table status registers (SYS_RAM_UNCERR_STSn)	R/W	0x0000_0000	*Section 53.6.2.9*
0x0434 + *n**0x8 (*n* = 0 to 1 - 1)	System RAM uncorrectable error reporting table address registers (SYS_RAM_UNCERR_ADDRn)	R/W	0x0000_0000	*Section 53.6.2.10*
0x0438 + (1 - 1)*0x8	System RAM uncorrectable error reporting table fill status register (SYS_RAM_UNCERR_TBL_FILL_STAT)	RO	0x0000_0000	*Section 53.6.2.11*
0x043C + (1 - 1)*0x8 - 0x063C	Reserved			
0x0640 + *n**0x4 (*n* = 0 to 45÷32 - 1)(1)	System RAM concurrent overflow registers(2) (SYS_RAM_OFLW*n*)	R/W	0x0000_0000	*Section 53.6.2.12*
0x0644 + (45÷32 - 1) *0x4 - 0x065C	Reserved			
0x0660 + n*0xC (n = 0 to 4 - 1)	System RAM ECC fault descriptor control registers (SYS_RAM_ECC_FD_CTRLn)	R/W	0x0000_0000	*Section 53.6.2.13*

Table 639. MEMU2 memory map (continued)

Offset or address	Register	Access	Reset value	Page
0x0664 + n*0xC (n = 0 to 4 - 1)	System RAM ECC fault descriptor start address registers (SYS_RAM_ECC_FD_STARTn)	R/W	0x0000_0000	*Section 53.6.2.14*
0x0668 + n*0xC (n = 0 to 4 - 1)	System RAM ECC fault descriptor end address registers (SYS_RAM_ECC_FD_ENDn)	R/W	0x0000_0000	*Section 53.6.2.15*
0x066C + (4 - 1)*0xC + n*0x4 (n = 0 to 45÷32-1)(1)	System RAM address extension registers (SYS_RAM_T32n)	R/W	0x0000_0000	*Section 53.6.2.16*
0x0670 + (4 - 1)*0xC + (45÷32 - 1)*0x4 - 0x0AFC	Reserved			
0x0B00 + *n*0x8 (*n* = 0 to 2 - 1)	Peripheral RAM correctable error reporting table status registers (PERIPH_RAM_CERR_STS*n*)	R/W	0x0000_0000	*Section 53.6.2.17*
0x0B04 + *n*0x8 (*n* = 0 to 2 - 1)	Peripheral RAM correctable error reporting table address registers (PERIPH_RAM_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 53.6.2.18*
0x0B08 + (2 - 1)*0x8	Peripheral RAM correctable error reporting table fill status register (PERIPH_RAM_CERR_TBL_FILL_STAT)	RO	0x0000_0000	*Section 53.6.2.19*
0x0B0C + (2 - 1)*0x8 - 0x0F0C	Reserved			
0x0F10 + *n*0x8 (*n* = 0 to 1 - 1)	Peripheral RAM uncorrectable error reporting table status registers (PERIPH_RAM_UNCERR_STSn)	R/W	0x0000_0000	*Section 53.6.2.20*
0x0F14 + *n*0x8 (*n* = 0 to 1 - 1)	Peripheral RAM uncorrectable error reporting table address registers (PERIPH_RAM_UNCERR_ADDRn)	R/W	0x0000_0000	*Section 53.6.2.21*
0x0F18 + (1 - 1)*0x8	Peripheral RAM uncorrectable error reporting table fill status register (PERIPH_RAM_UNCERR_TBL_FILL_STAT)	RO	0x0000_0000	*Section 53.6.2.22*
0x0F1C + (2 - 1)*0x8 - 0x111C	Reserved			
0x1120 + *n*0x4 (*n* = 0 to 38÷32 - 1)(1)	Peripheral RAM concurrent overflow registers(1) (PERIPH_RAM_OFLW*n*)	R/W	0x0000_0000	*Section 53.6.2.23*
0x1124 + (38÷32 - 1) *0x4 - 0x113C	Reserved			
0x1140 + n*0xC (n = 0 to 4 - 1)	Peripheral RAM ECC fault descriptor control registers (PERIPH_RAM_ECC_FD_CTRLn)	R/W	0x0000_0000	*Section 53.6.2.24*
0x1144 + n*0xC (n = 0 to 4 - 1)	Peripheral RAM ECC fault descriptor start address registers (PERIPH_RAM_ECC_FD_STARTn)	R/W	0x0000_0000	*Section 53.6.2.25*

Table 639. MEMU2 memory map (continued)

Offset or address	Register	Access	Reset value	Page
0x1148 + n*0xC (n = 0 to 4 - 1)	Peripheral RAM ECC fault descriptor end address registers (PERIPH_RAM_ECC_FD_ENDn)	R/W	0x0000_0000	*Section 53.6.2.26*
0x114C + (4 - 1)*0xC + n*0x4 (n = 0 to 38÷32 - 1)(1)	Peripheral RAM address extension registers (PERIPH_RAM_T32n)	R/W	0x0000_0000	*Section 53.6.2.27*
0x1150 + (4 - 1)*0xC + (38÷32 - 1)*0x4 - 0x15FC	Reserved			
0x1600 + *n*0x8 (*n* = 0 to 32 - 1)	NVM single-bit correctable error reporting table status registers (NVM_SB_CERR_STS*n*)	R/W	0x0000_0000	*Section 53.6.2.28*
0x1604 + *n*0x8 (*n* = 0 to 32 - 1)	NVM single-bit correctable error reporting table address registers (NVM_SB_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 53.6.2.29*
0x1608 + (32 - 1)*0x8	NVM single-bit correctable error reporting table fill status register (NVM_SB_CERR_TBL_FILL_STAT)	RO	0x0000_0000	*Section 53.6.2.30*
0x160C + (32 - 1)*0x8 - 0x1A0C	Reserved			
0x1A10 + *n*0x8 (*n* = 0 to 1 - 1)	NVM double-bit correctable error reporting table status registers (NVM_DB_CERR_STS*n*)	R/W	0x0000_0000	*Section 53.6.2.31*
0x1A14 + *n*0x8 (*n* = 0 to 1 - 1)	NVM double-bit correctable error reporting table address registers (NVM_DB_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 53.6.2.32*
0x1A18 + (1 - 1)*0x8	NVM double-bit correctable error reporting table fill status register (NVM_DB_CERR_TBL_FILL_STAT)	RO	0x0000_0000	*Section 53.6.2.33*
0x1A1C + (1 - 1)*0x8 - 0x1E1C	Reserved			
0x1E20 + *n*0x8 (n = 0 to 1 - 1)	NVM uncorrectable error reporting table status register (NVM_UNCERR_STS*n*)	R/W	0x0000_0000	*Section 53.6.2.34*
0x1E24 + *n*0x8 (n = 0 to 1 - 1)	NVM uncorrectable error reporting table address register (NVM_UNCERR_ADDR*n)*	R/W	0x0000_0000	*Section 53.6.2.35*
0x1E28 + (1 - 1)*0x8	NVM uncorrectable error reporting table fill status register (NVM_UNCERR_TBL_FILL_STAT)	RO	0x0000_0000	*Section 53.6.2.36*
0x1E2C + (1 - 1)*0x8 - 0x202C	Reserved			
0x2030 + *n*0x4 (*n* = 0 to 1÷32 - 1)(1)	NVM concurrent overflow registers(1) (NVM_OFLW*n*)	R/W	0x0000_0000	*Section 53.6.2.37*
0x2034 + (1÷32 - 1) *0x4 - 0x204C	Reserved			

Table 639. MEMU2 memory map (continued)

Offset or address	Register	Access	Reset value	Page
0x2050 + n*0xC (n = 0 to 4 - 1)	NVM ECC fault descriptor control registers (NVM_ECC_FD_CTRLn)	R/W	0x0000_0000	*Section 53.6.2.38*
0x2054 + n*0xC (n = 0 to 4 - 1)	NVM ECC fault descriptor start address registers (NVM_ECC_FD_STARTn)	R/W	0x0000_0000	*Section 53.6.2.39*
0x2058 + n*0xC (n = 0 to 4 - 1)	NVM ECC fault descriptor end address registers (NVM_ECC_FD_ENDn)	R/W	0x0000_0000	*Section 53.6.2.40*
0x205C + (4 - 1)*0xC + n*0x4 (n = 0 to 1÷32 - 1)(1)	NVM address extension registers (NVM_T32n)	R/W	0x0000_0000	*Section 53.6.2.41*
0x2060 + (4 - 1)*0xC + (1÷32 - 1)*0x4 - 0x3FFF	Reserved			

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.
2. The corresponding error sources for the Concurrent Overflow Register (OFLW) are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

53.6.2 Register descriptions

53.6.2.1 Control register (CTRL)

Offset: 0x0000 Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SWR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 904. Control register (CTRL)

Table 640. CTRL field descriptions

Field	Description
15 SWR	Software reset bit. It clears ERR_FLAG, SYS_RAM_OFLWn, PERIPH_RAM_OFLWn and NVM_RAM_OFLWn registers. This bit auto clears on next clock cycle. 0 No reset. 1 Reset asserted.

53.6.2.2 Error Flag register (ERR_FLAG)

Bits in the ERR_FLAG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_SCE, F_DCE, F_UCE, SR_CE, and SR_UCE)
- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCEO, SR_CEO, F_UCEO, F_DCEO, F_SCEO, PR_UCEO, and PR_CEO)

Individual flags can be cleared by writing a ‘1’. This also resets the corresponding error indication to FCCU for overflow flags. Flags do not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by software.

Bits SR_UCEO, SR_CEO, F_UCEO, F_SCEO, PR_UCEO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Offset: 0x0004 Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	PR_CE	PR_UCE	PR_CEO	PR_UCEO	0
W												w1c	w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	F_SCE	F_DCE	F_UCE	F_SCEO	F_DCEO	F_UCEO	0	0	0	0	SR_CE	SR_UCE	SR_CEO	SR_UCEO	0
W		w1c	w1c	w1c	w1c	w1c	w1c					w1c	w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 905. Error flag register (ERR_FLAG)

Table 641. ERR_FLAG field descriptions

Field	Description
20 PR_CE	Peripheral RAM ECC correctable error detect flag. Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
19 PR_UCE	Peripheral RAM ECC uncorrectable error detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.

Table 641. ERR_FLAG field descriptions (continued)

Field	Description
18 PR_CEO	Peripheral RAM ECC correctable error overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No overflow. 1 Overflow in the correctable error reporting table detected.
17 PR_UCEO	Peripheral RAM ECC uncorrectable error overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No overflow. 1 Overflow in the uncorrectable error reporting table detected.
14 F_SCE	NVM ECC single correctable error detect flag. Indicates that a new and unique NVM ECC single bit correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
13 F_DCE	NVM ECC double correctable error detect flag. Indicates that a new and unique NVM ECC double bit correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 F_UCE	NVM ECC uncorrectable error detect flag. Asserted when: – a new NVM ECC uncorrectable error was detected. – a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
11 F_SCEO	NVM ECC single correctable error overflow flag. Asserted when the NVM ECC single bit correctable error reporting table is full and a new entry is made to this table. 0 No overflow. 1 Overflow in the correctable error reporting table detected.
10 F_DCEO	NVM ECC double correctable error overflow flag. Asserted when the NVM ECC double bit correctable error reporting table is full and a new entry is made to this table. 0 No overflow. 1 Overflow in the correctable error reporting table detected.
9 F_UCEO	NVM ECC uncorrectable error overflow flag. Asserted when the NVM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No overflow. 1 Overflow in the uncorrectable error reporting table detected.

Table 641. ERR_FLAG field descriptions (continued)

Field	Description
4 SR_CE	System RAM ECC and MBIST correctable error detect flag. Indicates that a new and unique System RAM ECC and MBIST correctable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
3 SR_UCE	System RAM ECC and MBIST uncorrectable error detect flag. Asserted when: – A new System RAM ECC and MBIST uncorrectable error is detected – A new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
2 SR_CEO	System RAM ECC and MBIST correctable error overflow flag. Asserted when the System RAM ECC and MBIST correctable error reporting table is full and a new entry is made to this table. 0 No overflow. 1 Overflow in the correctable error reporting table detected.
1 SR_UCEO	System RAM ECC and MBIST uncorrectable error overflow flag. Asserted when the System RAM ECC and MBIST uncorrectable error reporting table is full and a new entry is made to this table. 0 No overflow. 1 Overflow in the uncorrectable error reporting table detected.

53.6.2.3 System RAM output trigger control register (SYS_RAM_OUT_TRIG_CTRL)

This register is used to check the connections between MEMU2 and FCCU via fake fault injection by software.

The error output of the MEMU2 towards FCCU stays set if forced until reset by software by writing 0s to this register.

Offset 0x000C Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FR_SR_FCCU_TRIG0	FR_SR_FCCU_TRIG1	FR_SR_FCCU_TRIG2	FR_SR_FCCU_TRIG3	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FR_SR_CEO	FR_SR_UCEO	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 906. System RAM output trigger control register (SYS_RAM_OUT_TRIG_CTRL)

Table 642. SYS_RAM_OUT_TRIG_CTRL field descriptions

Field	Description
31 FR_SR_FCCU_TRIG0	Forces FCCU_TRIG0 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.
30 FR_SR_FCCU_TRIG1	Forces FCCU_TRIG1 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.
29 FR_SR_FCCU_TRIG2	Forces FCCU_TRIG2 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.
28 FR_SR_FCCU_TRIG3	Forces FCCU_TRIG3 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.
2 FR_SR_CEO	Forces System RAM correctable error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag going towards FCCU to 1.
1 FR_SR_UCEO	Forces System RAM uncorrectable error overflow flag 0 Forcing is disabled. 1 Forces SR_UCEO flag going towards FCCU to 1.

53.6.2.4 Peripheral RAM output trigger control register (PERIPH_RAM_OUT_TRIG_CTRL)

This register is used to check the connections between MEMU2 and FCCU via fake fault injection by software.

The error output of the MEMU2 towards FCCU stays set if forced until reset by software via the MEMU2 in the normal way.

Offset 0x0010 Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FR_PR_CEO	FR_PR_UCEO	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	FR_PR_FCCU_TRIG3	FR_PR_FCCU_TRIG2	FR_PR_FCCU_TRIG1	FR_PR_FCCU_TRIG0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 907. Peripheral RAM output trigger control register (PERIPH_RAM_OUT_TRIG_CTRL)

Table 643. PERIPH_RAM_OUT_TRIG_CTRL field descriptions

Field	Description
18 FR_PR_CEO	Force Peripheral RAM correctable error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag going towards FCCU to 1.
17 FR_PR_UCEO	Forces Peripheral RAM uncorrectable error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCEO flag going towards FCCU to 1.
3 FR_PR_FCCU_TRIG3	Forces FCCU_TRIG3 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.
2 FR_PR_FCCU_TRIG2	Forces FCCU_TRIG2 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.

Table 643. PERIPH_RAM_OUT_TRIG_CTRL field descriptions (continued)

Field	Description
1 FR_PR_FCCU_TRIG1	Forces FCCU_TRIG1 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.
0 FR_PR_FCCU_TRIG0	Forces FCCU_TRIG0 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.

53.6.2.5 NVM output trigger control register (NVM_OUT_TRIG_CTRL)

This register is used to check the connections between MEMU2 and FCCU via fake fault injection by software.

The error output of the MEMU2 towards FCCU stays set if forced until reset by software via the MEMU2 in the normal way.

Offset 0x0014 Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	FR_F_FCCU_TRIG0	FR_F_FCCU_TRIG1	FR_F_FCCU_TRIG2	FR_F_FCCU_TRIG3												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	FR_F_SCEO	FR_F_UCEO	FR_F_DCEO	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 908. NVM output trigger control register (NVM_OUT_TRIG_CTRL)

Table 644. NVM_OUT_TRIG_CTRL field descriptions

Field	Description
31 FR_F_FCCU_TRIG0	Forces FCCU_TRIG0 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.
30 FR_F_FCCU_TRIG1	Forces FCCU_TRIG1 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.

Table 644. NVM_OUT_TRIG_CTRL field descriptions (continued)

Field	Description
29 FR_F_FCCU_TRIG2	Forces FCCU_TRIG2 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.
28 FR_F_FCCU_TRIG3	Forces FCCU_TRIG3 to 1 0 Forcing is disabled. 1 Forces trigger going to FCCU to 1.
10 FR_F_SCEO	Forces NVM single correctable error overflow flag 0 Forcing is disabled. 1 Forces F_SCEO flag going towards FCCU to 1.
9 FR_F_UCEO	Forces NVM uncorrectable error overflow flag 0 Forcing is disabled. 1 Forces F_UCEO flag going towards FCCU to 1.
8 FR_F_DCEO	Forces NVM double correctable error overflow flag 0 Forcing is disabled. 1 Forces F_DCEO flag going towards FCCU to 1.

53.6.2.6 System RAM correctable error reporting table status register (SYS_RAM_CERR_STS*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0020 + *n**0x8 (*n* = 0 to (10 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VLD	0	0	ECC_OR_MBIST	0	0	0	0	MEM_ID							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 909. System RAM correctable error reporting table status register (SYS_RAM_CERR_STS*n*)

Table 645. SYS_RAM_CERR_STS*n* field descriptions

Field	Description
31 VLD	Valid bit This indicates that the entry in reporting table is valid. 0 entry is not valid. 1 entry is valid.
28 ECC_OR_MBIST	This field indicates whether the error logged in the reporting table entry is an ECC or MBIST error. 0 MBIST error. 1 ECC error.
23:16 MEM_ID	This field indicates the error source index corresponding to the BAD_BIT captured. The 6 MSB bits indicate 1 of the 64 MEMU channels while the 2 LSBs indicate one of the 4 sources connected to a single MEMU channel. **Note:** The bits within the MEM_ID field are not intended for user access. Users must consider them RESERVED.
7:0 BAD_BIT	Bad bit field For ECC errors, this field indicates failed bit syndrome. For ECC errors whose memory sources do not generate any syndrome, this field is read as FF. For MBIST errors, this field indicates bit position in RAM where the error is detected.

53.6.2.7 System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR*n*)

Offset: 0x0024 + *n**0x8 (*n* = 0 to (10 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 910. System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR*n*)

Table 646. SYS_RAM_CERR_ADDR*n* field descriptions

Field	Description
31:0 ERR_ADD	Error address field Indicates the address on which the error was detected. Only word access for write is allowed, other types of access do not result in a successful write to this register.

53.6.2.8 System RAM correctable error reporting table fill status register (SYS_RAM_CERR_TBL_FILL_STAT)

Offset: 0x0028 + (10 - 1)*0x8 Access: RO

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	CERR_TBL_FILL_VAL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 911. System RAM correctable error reporting table fill status register (SYS_RAM_CERR_TBL_FILL_STAT)

Table 647. SYS_RAM_CERR_TBL_FILL_STAT field descriptions

Field	Description
7:0 CERR_TBL_FILL_VAL	This value indicates the number of valid entries in the correctable table for System RAM

53.6.2.9 System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STSn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0430 + *n**0x8 (*n* = 0 to (1 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 912. System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STSn)

Table 648. SYS_RAM_UNCERR_STSn field descriptions

Field	Description
31 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 entry is not valid. 1 entry is valid.

53.6.2.10 System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDRn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0434 + *n**0x8 (*n* = 0 to (1 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 913. System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDRn)

Table 649. SYS_RAM_UNCERR_ADDRn field descriptions

Field	Description
31:0 ERR_ADD	Error address field Indicates the address on which the error was detected.Only word access for write is allowed, other types of access do not result in a successful write to this register.

53.6.2.11 System RAM uncorrectable error reporting table fill status register (SYS_RAM_UNCERR_TBL_FILL_STAT)

Offset: 0x0438 + (1 - 1)*0x8 Access: RO

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	UNCERR_TBL_FILL_VAL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 914. System RAM uncorrectable error reporting table fill status register (SYS_RAM_UNCERR_TBL_FILL_STAT)

Table 650. SYS_RAM_UNCERR_TBL_FILL_STAT field descriptions

Field	Description
7:0 UNCERR_TBL_FILL_VAL	This value indicates the number of valid entries in the uncorrectable table for System RAM

53.6.2.12 System RAM concurrent overflow registers (SYS_RAM_OFLW*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0640 + *n**0x4 (*n* = 0 to (45÷32 - 1))(1) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 915. System RAM concurrent overflow registers (SYS_RAM_OFLW*n*)

Table 651. SYS_RAM_OFLW*n* field descriptions

Field	Description
31:0 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 53.7.3: Handling overflows* occurs.

53.6.2.13 System RAM ECC fault descriptor control registers (SYS_RAM_ECC_FD_CTRL*n*)

This register is used to configure the ECC fault descriptor.

Offset: 0x0660 + *n**0xC (*n* = 0 to (4 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN	0	0	0	0	0	0	0	0	0	0	0	0	UC	0	SBC
W																
Reset	0/1(1)	0	0	0	0	0	0	0	0	0	0	0	0	0/1(1)	0	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	T32	0	0	0	0	0	0	0	0	0	0	0	FCCU_TRG			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 1 for n=0, 0 for n>0

Figure 916. System RAM ECC fault descriptor control registers (SYS_RAM_ECC_FD_CTRL*n*)

Table 652. SYS_RAM_ECC_FD_CTRL*n* field descriptions

Field	Description
31 EN	0 ECC fault descriptor must not be used 1 ECC fault descriptor must be used
18 UC	0 Uncorrectable ECC errors must be ignored 1 Uncorrectable ECC errors must be taken in account
16 SBC	0 Single-bit ECC errors must be ignored 1 Single-bit ECC errors must be taken in account
15 T32	0 addresses used for the comparison are the ones specified in START_ADR and END_ADR registers 1 addresses used for the comparison are the ones specified in START_ADR and END_ADR registers + 2^{32}
3:0 FCCU_TRG	0x0: SYS_RAM_TRIG_0 is asserted 0x1: SYS_RAM_TRIG_1 is asserted 0x2: SYS_RAM_TRIG_2 is asserted 0x3: SYS_RAM_TRIG_3 is asserted 0x4 to 0xE: reserved 0xF: no FCCU trigger is asserted

53.6.2.14 System RAM ECC fault descriptor start address registers (SYS_RAM_ECC_FD_START*n*)

This register is used to configure the region start address.

Offset: 0x0664 + *n**0xC (*n* = 0 to (4 - 1)) Access: R/W

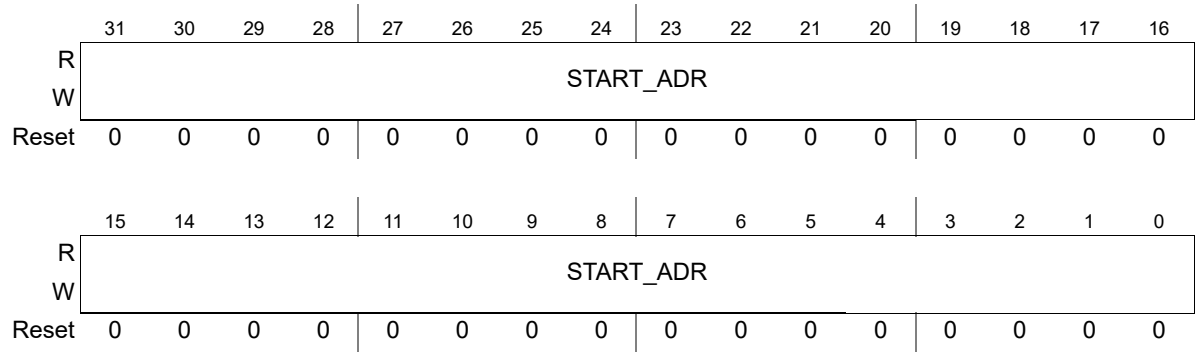


Figure 917. System RAM ECC fault descriptor start address registers (SYS_RAM_ECC_FD_START*n*)

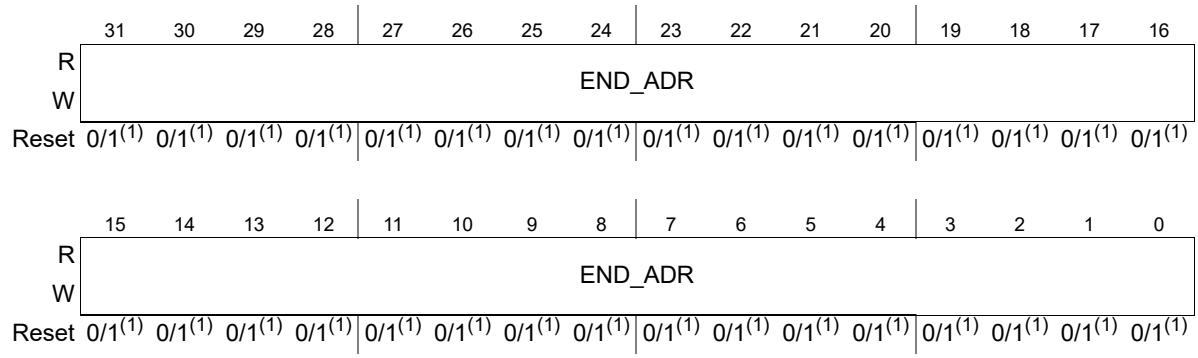
Table 653. SYS_RAM_ECC_FD_START*n* field descriptions

Field	Description
31:0 START_ADR	Address indicating the region start address. Only word access for write is allowed, other types of access do not result in a successful write to this register.

53.6.2.15 System RAM ECC fault descriptor end address registers (SYS_RAM_ECC_FD_END*n*)

This register is used to configure the region end address.

Offset: 0x0668 + *n**0xC (*n* = 0 to (4 - 1)) Access: R/W



1. 1 for n=0, 0 for n>0

Figure 918. System RAM ECC fault descriptor end address registers (SYS_RAM_ECC_FD_END*n*)

Table 654. SYS_RAM_ECC_FD_END*n* field descriptions

Field	Description
31:0 END_ADR	Address indicating the region end address. Only word access for write is allowed, other types of access do not result in a successful write to this register.

53.6.2.16 System RAM address extension registers (SYS_RAM_T32*n*)

These registers are used to enable a 33rd address bit as MSB bit of the address used for the ECC fault filtering.

The bits must be set in a set of “n” registers, following the order to the MEMU2 ECC input fault channel, that is, bit 0 of register 0 is used to extend the address reported in the MEMU2 ECC input fault channel 0 + n*32, and so on.

Offset: 0x066C + (4 - 1)*0xC + *n**0x4 (*n* = 0 to (45÷32)-1)(1) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 919. System RAM address extension registers (SYS_RAM_T32*n*)

Table 655. SYS_RAM_T32*n* field descriptions

Field	Description
31:0 T[31:0]	MSB bit to be added to extend the reported address of the MEMU2 input ECC fault channel n*32 + i (where i is the bit number of the corresponding bit field Ti).

53.6.2.17 Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STSn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0B00 + *n**0x8 (*n* = 0 to (2 - 1)) Access: R/W

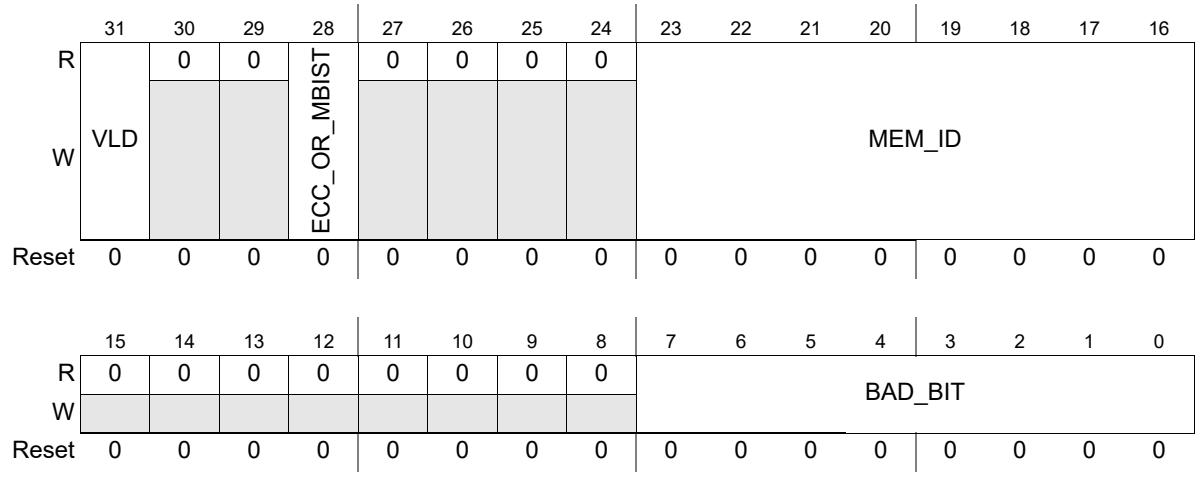


Figure 920. Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS*n*)

Table 656. PERIPH_RAM_CERR_STS*n* field descriptions

Field	Description
31 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 entry is not valid. 1 entry is valid.
28 ECC_OR_MBIST	This field indicates whether the error logged in the reporting table entry is an ECC or MBIST error. 0 MBIST error. 1 ECC error.
23:16 MEM_ID	This field indicates the error source index corresponding to the BAD_BIT captured. The 6 MSB bits indicate 1 of the 64 MEMU channels while the 2 LSBs indicate one of the 4 sources connected to a single MEMU channel. **Note:** The bits within the MEM_ID field are not intended for user access. Users must consider them RESERVED.
7:0 BAD_BIT	Bad bit field For ECC errors, this field indicates failed bit syndrome. For ECC errors whose memory sources do not generate any syndrome, this field is read as FF. For MBIST errors, this field indicates bit position in RAM where the error is detected.

53.6.2.18 Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR*n*)

Offset: 0x0B04 + *n**0x8 (*n* = 0 to (2 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 921. Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR*n*)

Table 657. PERIPH_RAM_CERR_ADDR*n* field descriptions

Field	Description
31:0 ERR_ADD	Error address field Indicates the address on which the error was detected. Only word access for write is allowed, other types of access do not result in a successful write to this register.

53.6.2.19 Peripheral RAM correctable error reporting table fill status register (PERIPH_RAM_CERR_TBL_FILL_STAT)

Offset: 0x0B08 + (2 - 1)*0x8 Access: RO

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	CERR_TBL_FILL_VAL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 922. Peripheral RAM correctable error reporting table fill status register (PERIPH_RAM_CERR_TBL_FILL_STAT)

Table 658. PERIPH_RAM_CERR_TBL_FILL_STAT field descriptions

Field	Description
7:0 CERR_TBL_FILL_VAL	This value indicates the number of valid entries in the correctable table for Peripheral RAM.

53.6.2.20 Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STSn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0F10 + *n**0x8 (*n* = 0 to (1 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 923. Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STSn)

Table 659. PERIPH_RAM_UNCERR_STSn field descriptions

Field	Description
31 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 entry is not valid. 1 entry is valid.

53.6.2.21 Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDRn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0F14 + *n**0x8 (*n* = 0 to (1 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 924. Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDRn)

Table 660. PERIPH_RAM_UNCERR_ADDRn field descriptions

Field	Description
31:0 ERR_ADD	Error address field Indicates the address on which the error was detected. Only word access for write is allowed, other types of access do not result in a successful write to this register.

53.6.2.22 Peripheral RAM uncorrectable error reporting table fill status register (PERIPH_RAM_UNCERR_TBL_FILL_STAT)

Offset: 0x0F18 + (1 - 1)*0x8 Access: RO

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	UNCERR_TBL_FILL_VAL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 925. Peripheral RAM uncorrectable error reporting table fill status register (PERIPH_RAM_UNCERR_TBL_FILL_STAT)

Table 661. PERIPH_RAM_UNCERR_TBL_FILL_STAT field descriptions

Field	Description
7:0 UNCERR_TBL_FILL_VAL	This value indicates the number of valid entries in the uncorrectable table for Peripheral RAM.

53.6.2.23 Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLW*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x1120 + *n**0x4 (*n* = 0 to (38÷32 - 1))(1) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 926. Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLW*n*)

Table 662. PERIPH_RAM_OFLW*n* field descriptions

Field	Description
31:0 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 53.7.3: Handling overflows* occurs.

53.6.2.24 Peripheral RAM ECC fault descriptor control registers (PERIPH_RAM_ECC_FD_CTRL*n*)

This register is used to configure the ECC fault descriptor.

Offset: 0x1140 + *n**0xC (*n* = 0 to (4 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN	0	0	0	0	0	0	0	0	0	0	0	0	UC	0	SBC
W																
Reset	0/1(1)	0	0	0	0	0	0	0	0	0	0	0	0	0/1(1)	0	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	T32	0	0	0	0	0	0	0	0	0	0	0	FCCU_TRG			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 1 for n=0, 0 for n>0

Figure 927. Peripheral RAM ECC fault descriptor control registers (PERIPH_RAM_ECC_FD_CTRL*n*)

Table 663. PERIPH_RAM_ECC_FD_CTRL*n* field descriptions

Field	Description
31 EN	0 ECC fault descriptor must not be used 1 ECC fault descriptor must be used
18 UC	0 Uncorrectable ECC errors must be ignored 1 Uncorrectable ECC errors must be taken in account
16 SBC	0 Single-bit ECC errors must be ignored 1 Single-bit ECC errors must be taken in account
15 T32	0 addresses used for the comparison are the ones specified in START_ADR and END_ADR registers 1 addresses used for the comparison are the ones specified in START_ADR and END_ADR registers + 2^{32}
3:0 FCCU_TRG	0x0: PERIPH_RAM_TRIG_0 is asserted 0x1: PERIPH_RAM_TRIG_1 is asserted 0x2: PERIPH_RAM_TRIG_2 is asserted 0x3: PERIPH_RAM_TRIG_3 is asserted 0x4 to 0xE: reserved 0xF: no FCCU trigger is asserted

53.6.2.25 Peripheral RAM ECC fault descriptor start address registers (PERIPH_RAM_ECC_FD_START*n*)

This register is used to configure the region start address.

Offset: 0x1144 + *n**0xC (*n* = 0 to (4 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	START_ADR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	START_ADR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 928. Peripheral RAM ECC fault descriptor start address registers (PERIPH_RAM_ECC_FD_START*n*)

Table 664. PERIPH_RAM_ECC_FD_START*n* field descriptions

Field	Description
31:0 START_ADR	Address indicating the region start address. Only word access for write is allowed, other types of access do not result in a successful write to this register.

53.6.2.26 Peripheral RAM ECC fault descriptor end address registers (PERIPH_RAM_ECC_FD_END*n*)

This register is used to configure the region end address.

Offset: 0x1148 + *n**0xC (*n* = 0 to (4 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	END_ADR															
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	END_ADR															
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. 1 for n=0, 0 for n>0

Figure 929. Peripheral RAM ECC fault descriptor end address registers (PERIPH_RAM_ECC_FD_END*n*)

Table 665. PERIPH_RAM_ECC_FD_END*n* field descriptions

Field	Description
31:0 END_ADR	Address indicating the region end address. Only word access for write is allowed, other types of access do not result in a successful write to this register.

53.6.2.27 Peripheral RAM address extension registers (PERIPH_RAM_T32*n*)

These registers are used to enable a 33rd address bit as MSB bit of the address used for the ECC fault filtering.

The bits must be set in a set of “n” registers, following the order to the MEMU2 ECC input fault channel, that is, bit 0 of register 0 is used to extend the address reported in the MEMU2 ECC input fault channel 0 + n*32, and so on.

Offset: 0x114C + (4 - 1)*0xC + *n**0x4 (*n* = 0 to (38÷32)-1)(1) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 930. Peripheral RAM address extension registers (PERIPH_RAM_T32*n*)

Table 666. PERIPH_RAM_T32*n* field descriptions

Field	Description
31:0 T[31:0]	MSB bit to be added to extend the reported address of the MEMU2 input ECC fault channel n*32 + i (where i is the bit number of the corresponding bit field Ti).

53.6.2.28 NVM single-bit correctable error reporting table status register (NVM_SB_CERR_STS*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x1600 + *n**0x8 (*n* = 0 to (32 - 1)) Access: R/W

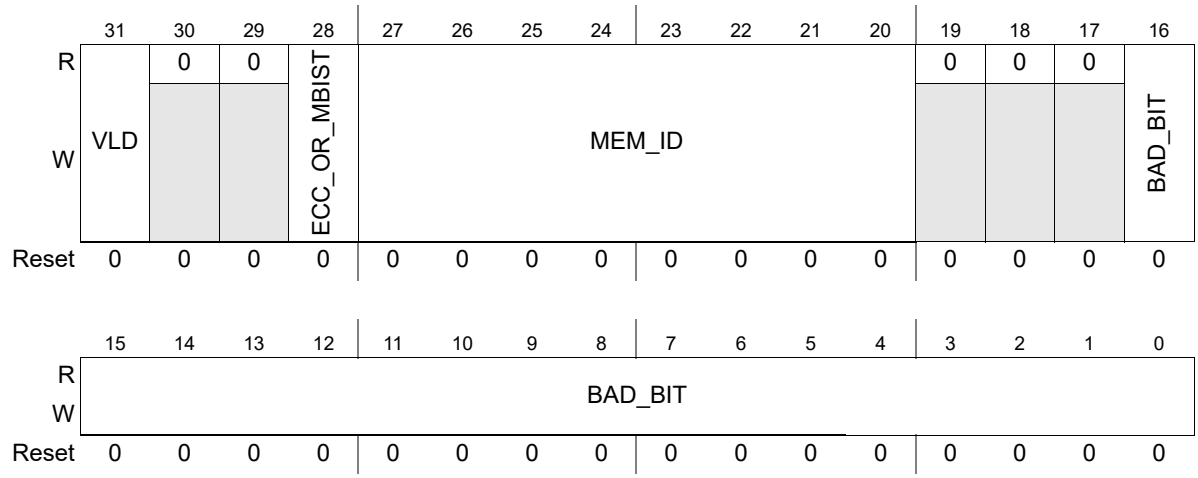


Figure 931. NVM single-bit correctable error reporting table status register (NVM_SB_CERR_STS*n*)

Table 667. NVM_SB_CERR_STS*n* register field descriptions

Field	Description
31 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 entry is not valid. 1 entry is valid.
28 ECC_OR_MBIST	This field indicates whether the error logged in the reporting table entry is an ECC or MBIST error. 0 MBIST error. 1 ECC error. **Note:** Since MBIST errors are not reported on NVM, this field is always read as 1, whenever a new error is stored. Software must also program it as 1, whenever directly storing an entry in the reporting table.

Table 667. NVM_SB_CERR_STS*n* register field descriptions (continued)

Field	Description
27:20 MEM_ID	This field indicates the error source index corresponding to the BAD_BIT captured. The 6 MSB bits indicate 1 of the 64 MEMU channels while the 2 LSBs indicate one of the 4 sources connected to a single MEMU channel. **Note:** The bits within the MEM_ID field are not intended for user access. Users must consider them RESERVED.
16:0 BAD_BIT	Bad bit field For ECC errors, this field indicates failed bit syndrome. For ECC errors whose memory sources do not generate any syndrome, this field is read as 1FFFF. For MBIST errors, this field indicates bit position where the error is detected. **Note:** Byte or half-word access is not allowed. It leads to an unsuccessful write on this field.

53.6.2.29 NVM single-bit correctable error reporting table address register (NVM_SB_CERR_ADDR*n*)

Offset: 0x1604 + *n**0x8 (*n* = 0 to (32 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 932. NVM single-bit correctable error reporting table address register (NVM_SB_CERR_ADDR*n*)

Table 668. NVM_SB_CERR_ADDR*n* register field descriptions

Field	Description
31:0 ERR_ADD	Error address field Indicates the address on which the error was detected. Only word access for write is allowed, other types of software access do not result in a successful write to this register.

53.6.2.30 NVM single-bit correctable error reporting table fill status register (NVM_SB_CERR_TBL_FILL_STAT)

Offset: 0x1608 + (32 - 1)*0x8 Access: RO

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	SB_CERR_TBL_FILL_VAL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 933. NVM single-bit correctable error reporting table fill status register (NVM_SB_CERR_TBL_FILL_STAT)

Table 669. NVM_SB_CERR_TBL_FILL_STAT field descriptions

Field	Description
7:0 SB_CERR_TBL_FILL_VAL	This value indicates the number of valid entries in the single correctable table for NVM

53.6.2.31 NVM double-bit correctable error reporting table status register (NVM_DB_CERR_STS*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x1A10 + *n**0x8 (*n* = 0 to (1 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		0	0										0	0	0	
W	VLD			ECC_OR_MBIST	MEM_ID											BAD_BIT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BAD_BIT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 934. NVM double-bit correctable error reporting table status register (NVM_DB_CERR_STS*n*)

Table 670. NVM_DB_CERR_STS*n* register field descriptions

Field	Description
31 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 entry is not valid. 1 entry is valid.
28 ECC_OR_MBIST	This field indicates whether the error logged in the reporting table entry is an ECC or MBIST error. 0 MBIST error. 1 ECC error. **Note:** Since MBIST errors are not reported on NVM, this field is always read as 1, whenever a new error is stored. Software must also program it as 1, whenever directly storing an entry in the reporting table.
27:20 MEM_ID	This field indicates the channel index corresponding to the BAD_BIT captured. The 6 MSB bits indicate 1 of the 64 MEMU channels while the 2 LSBs indicate one of the 4 sources connected to a single MEMU channel. **Note:** The bits within the MEM_ID field are not intended for user access. Users must consider them RESERVED.
16:0 BAD_BIT	Bad bit field For ECC errors, this field indicates failed bit syndrome. For ECC errors whose memory sources do not generate any syndrome, this field is read as 1FFFF. For MBIST errors, this field indicates bit position where the error is detected. **Note:** Byte or half-word access is not allowed. It leads to an unsuccessful write on this field.

53.6.2.32 NVM double-bit correctable error reporting table address register (NVM_DB_CERR_ADDR*n*)

Offset: 0x1A14 + *n**0x8 (*n* = 0 to (1 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 935. NVM double-bit correctable error reporting table address register (NVM_DB_CERR_ADDR*n*)

Table 671. NVM_DB_CERR_ADDR*n* register field descriptions

Field	Description
31:0 ERR_ADD	Error address field Indicates the address on which the error was detected.Only word access for write is allowed, other types of software access do not result in a successful write to this register.

53.6.2.33 NVM double-bit correctable error reporting table fill status register (NVM_DB_CERR_TBL_FILL_STAT)

Offset: 0x1A18 + (1 - 1)*0x8 Access: RO

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	DB_CERR_TBL_FILL_VAL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 936. NVM double-bit correctable error reporting table fill status register (NVM_DB_CERR_TBL_FILL_STAT)

Table 672. NVM_DB_CERR_TBL_FILL_STAT field descriptions

Field	Description
7:0 DB_CERR_TBL_FILL_VAL	This value indicates the number of valid entries in the double correctable table for NVM.

53.6.2.34 NVM uncorrectable error reporting table status register (NVM_UNCERR_STS*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x1E20 + *n**0x8 (n = 0 to 1 - 1) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 937. NVM uncorrectable error reporting table status register (NVM_UNCERR_STS*n*)

Table 673. NVM_UNCERR_STS*n* field descriptions

Field	Description
31 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 entry is not valid. 1 entry is valid.

53.6.2.35 NVM uncorrectable error reporting table address register (NVM_UNCERR_ADDR*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x1E24 + *n**0x8 (n = 0 to 1 - 1) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 938. NVM uncorrectable error reporting table address register (NVM_UNCERR_ADDR*n*)

Table 674. NVM_UNCERR_ADDR*n* field descriptions

Field	Description
31:0 ERR_ADD	Error address field Indicates the address on which the error was detected. Only word access for write is allowed, other types of software access do not result in a successful write to this register.

53.6.2.36 NVM uncorrectable error reporting table fill status register (NVM_UNCERR_TBL_FILL_STAT)

Offset: 0x1E28 + (1 - 1)*0x8 Access: RO

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	UNCERR_TBL_FILL_VAL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 939. NVM uncorrectable error reporting table fill status register (NVM_UNCERR_TBL_FILL_STAT)

Table 675. NVM_UNCERR_TBL_FILL_STAT field descriptions

Field	Description
7:0 UNCERR_TBL_FILL_VAL	This value indicates the number of valid entries in the uncorrectable table for NVM.

53.6.2.37 NVM concurrent overflow registers (NVM_OFLWn)

Note: The reporting table is not cleared on software reset.

Offset: 0x2030 + *n**0x4 (*n* = 0 to (1÷32 - 1))(1) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 940. NVM concurrent overflow registers (NVM_OFLWn)

Table 676. NVM_OFLWn field descriptions

Field	Description
31:0 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 53.7.3: Handling overflows* occurs.

53.6.2.38 NVM ECC fault descriptor control registers (NVM_ECC_FD_CTRL*n*)

This register is used to configure the ECC fault descriptor.

Offset: 0x2050 + *n**0xC (*n* = 0 to (4 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EN	0	0	0	0	0	0	0	0	0	0	0	0	UC	DBC	SBC
W																
Reset	0/1(1)	0	0	0	0	0	0	0	0	0	0	0	0	0/1(1)	0/1(1)	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	T32	0	0	0	0	0	0	0	0	0	0	0	FCCU_TRG			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 1 for n=0, 0 for n>0

Figure 941. NVM ECC fault descriptor control registers (NVM_ECC_FD_CTRL*n*)

Table 677. NVM_ECC_FD_CTRL*n* field descriptions

Field	Description
31 EN	0 ECC fault descriptor must not be used 1 ECC fault descriptor must be used
18 UC	0 Uncorrectable ECC errors must be ignored 1 Uncorrectable ECC errors must be taken in account
17 DBC	0 Double-bit ECC errors must be ignored 1 Double-bit ECC errors must be taken in account
16 SBC	0 Single-bit ECC errors must be ignored 1 Single-bit ECC errors must be taken in account
15 T32	0 addresses used for the comparison are the ones specified in START_ADR and END_ADR registers 1 addresses used for the comparison are the ones specified in START_ADR and END_ADR registers + 2^{32}
3:0 FCCU_TRG	0x0: NVM_TRIG_0 is asserted 0x1: NVM_TRIG_1 is asserted 0x2: NVM_TRIG_2 is asserted 0x3: NVM_TRIG_3 is asserted 0x4 to 0xE: reserved 0xF: no FCCU trigger is asserted

53.6.2.39 NVM ECC fault descriptor start address registers (NVM_ECC_FD_START*n*)

This register is used to configure the region start address.

Offset: 0x2054 + *n**0xC (*n* = 0 to (4 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	START_ADR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	START_ADR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 942. NVM ECC fault descriptor start address registers (NVM_ECC_FD_START*n*)

Table 678. NVM_ECC_FD_START*n* field descriptions

Field	Description
31:0 START_ADR	Address indicating the region start address. Only word access for write is allowed, other types of software access do not result in a successful write to this register.

53.6.2.40 NVM ECC fault descriptor end address registers (NVM_ECC_FD_END*n*)

This register is used to configure the region end address.

Offset: 0x2058 + *n**0xC (*n* = 0 to (4 - 1)) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	END_ADR															
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	END_ADR															
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. 1 for n=0, 0 for n>0

Figure 943. NVM ECC fault descriptor end address registers (NVM_ECC_FD_END*n*)

Table 679. NVM_ECC_FD_END*n* field descriptions

Field	Description
31:0 END_ADR	Address indicating the region end address. Only word access for write is allowed, other types of software access do not result in a successful write to this register.

53.6.2.41 NVM address extension registers (NVM_T32*n*)

These registers are used to enable a 33rd address bit as MSB bit of the address used for the ECC fault filtering.

The bits must be set in a set of “n” registers, following the order to the MEMU2 ECC input fault channel, that is, bit 0 of register 0 is used to extend the address reported in the MEMU2 ECC input fault channel 0 + n*32, and so on.

Offset: 0x205C + (4 - 1)*0xC + *n**0x4 (*n* = 0 to (1÷32)-1)(1) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 944. NVM address extension registers (NVM_T32*n*)

Table 680. NVM_T32*n* field descriptions

Field	Description
31:0 T[31:0]	MSB bit to be added to extend the reported address of the MEMU2 input ECC fault channel n*32 + i (where i is the bit number of the corresponding bit field Ti).

53.7 Functional description

In this section, the terms “bad bit” and “syndrome” are equivalent, and are used to indicate the position of the bit in error.

53.7.1 Initializing MEMU2

MEMU2 is always enabled and can be configured by user software for known errors which the system/application has collected over a period of time. The MEMU2 logic does not update the reporting table corresponding to the programmed values which have the VLD bit set.

The user software can write into the reporting table and must make VLD bit = 1’b1 for the programmed entry to be valid.

The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address.The sequence of programming to prevent a collision of reporting table entries written by the hardware and the software is as follows:

1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

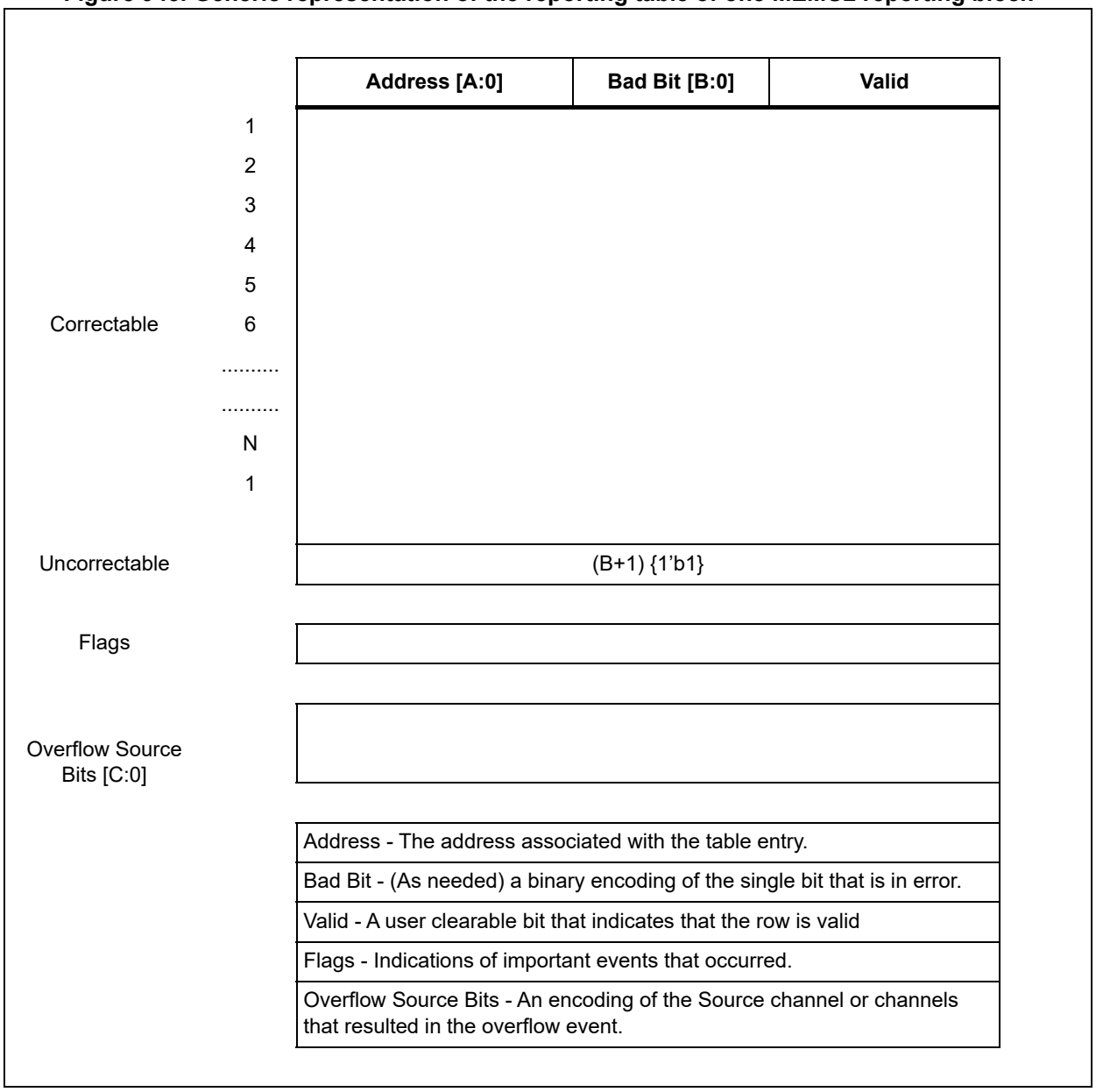
If the FCCU is programmed to cause an IRQ on new MEMU2 table entries, steps 3 to 5 must be executed with interrupts disabled to prevent any error handling software to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 5th step, and the MEMU2 writes an error with the same address that software wants to program, both the errors are stored in the reporting table (duplicate entries).

53.7.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It reads the entries with the valid bit set and takes the necessary action. To invalidate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set returns invalid data.

Based on MBIST implementation, the MBIST might report several correctable errors in the same word instead of a non-correctable one. It is the task of the software after MBIST to aggregate these reports and detect the non-correctable error: this happens because the MBIST logic reports to the MEMU each failed bit as a separate ECC failure, indicating in the syndrome the index of the failed bit.

Figure 945. Generic representation of the reporting table of one MEMU2 reporting block



53.7.3 Handling overflows

When a bit in the overflow registers is asserted it can indicate the occurrence of the following condition: overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).

54 Indirect memory access (IMA)

54.1 Introduction

Indirect memory access (IMA) refers to the activity of accessing any chip memory for the purpose of reading or modifying data, or both, including ECC check bits. This capability is useful for test activities, for example, verifying ECC functionality.

On SR5E1x MCU, the IMA controller module provides registers for selecting, reading, and writing memory data.

Note: Not all SRAM data and ECC bits are accessible. Refer to the Device Configuration chapter for details.

54.2 Accessing memory through the IMA module

The Indirect Memory Access (IMA) module provides an alternative way to access on-chip memory without going through standard peripheral or system RAM interfaces to check the data or change the data.

This alternative path can be used for diagnostic purposes. For example, for SRAM arrays that do not support E2E ECC, the IMA block provides a way to write corrupted ECC values so that the ECC error decoding logic can be checked.

Note the following restrictions on memory access via IMA:

- When a bus master performs an IMA access, no other bus master must perform an access to the same memory.
- Before any IMA access, the user must disable the FCCU fault related to the IMA. If not, the FCCU triggers a fault.
- The IMA must be unlocked for reads.

The order of precedence for access to a memory is shown below. Once the IMA has been granted access, no other accesses are allowed until the IMA select line to the selected memory is negated by the IMA.

Note: Once the IMA has selected a RAM, it expects no other accesses from any other master until deselected by the IMA.

The IMA module signals the FCCU when it is reading or writing to a memory address. This enables the FCCU to assess whether the access is intentional or spurious. For an intentional access the respective FCCU error input needs to be disabled, otherwise it must be enabled.

54.2.1 Features

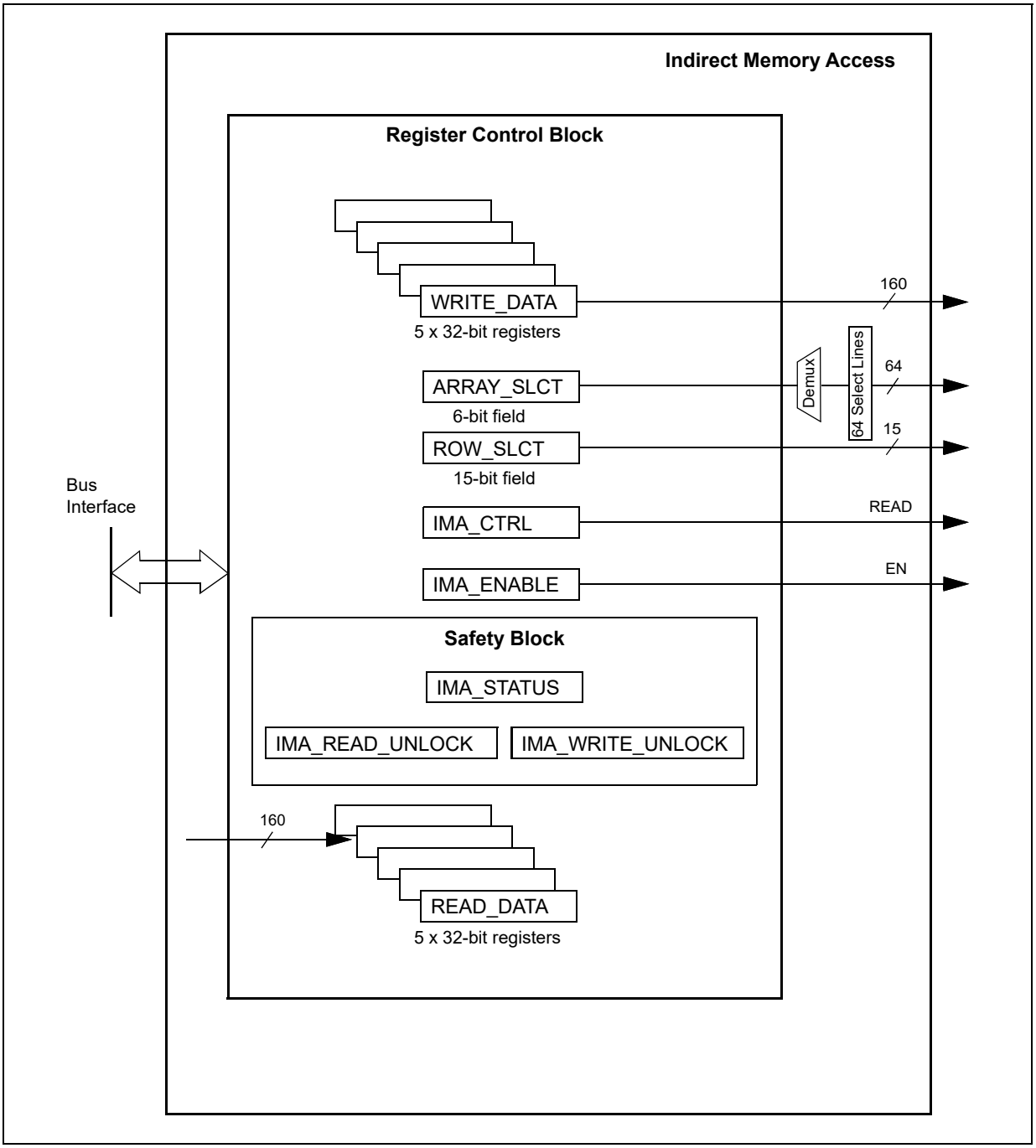
The IMA module includes the following features:

- Able to read all bits in all RAM arrays connected to the IMA, once the IMA is unlocked for reads.
- Enables reading and manipulation of ECC check bits.
- During an IMA SRAM access, other accesses are affected as follows:
  - Reads return random data
  - Writes have no effect
- Any IMA access to memory are indicated in the Fault Collection and Control Unit (FCCU).
- IMA accesses to memory do not bypass any SRAM repair which may have occurred during production test of the device.
- IMA accesses to memory use a physical address, not a logical address, and thus memory addresses used by the IMA are not memory mapped.

54.2.2 Block diagram

The following figure shows the structure and function of the IMA module. The Register Control Block contains the memory-mapped registers accessible by software. The Safety Block prevents spurious activation of the IMA module. It ensures the IMA does not access RAM unless the system needs it too.

Figure 946. IMA block diagram



54.2.3 IMA module memory map and registers

This section provides details of the IMA memory map. It also provides detailed descriptions for each of the registers in that map.

The Indirect Memory Access unit's memory map allows for the CPU to access RAM's on the chip to write test data so that single-bit ECCs and multi-bit ECC logic can be tested.

Table 681. IMA memory map

Offset Address	Register	Section
0x0000	IMA Control register (IMA_CTRL)	*Section 54.2.3.1*
0x0004	IMA Enable Access register (IMA_ENABLE)	*Section 54.2.3.2*
0x0008	IMA Status register (IMA_STATUS)	*Section 54.2.3.3*
0x000C	IMA RAM SELECT register (IMA_SLCT)	*Section 54.2.3.4*
0x0010	IMA Write Unlock register (IMA_WRITE_UNLOCK)	*Section 54.2.3.5*
0x0014	IMA Read Unlock register (IMA_READ_UNLOCK)	*Section 54.2.3.6*
0x0018–0x002B	Reserved	
0x002C	IMA RAM Write Data register 4 (IMA_WRITE_DATA_4)	*Section 54.2.3.7*
0x0030	IMA RAM Write Data register 3 (IMA_WRITE_DATA_3)	*Section 54.2.3.7*
0x0034	IMA RAM Write Data register 2 (IMA_WRITE_DATA_2)	*Section 54.2.3.7*
0x0038	IMA RAM Write Data register 1 (IMA_WRITE_DATA_1)	*Section 54.2.3.7*
0x003C	IMA RAM Write Data register 0 (IMA_WRITE_DATA_0)	*Section 54.2.3.7*
0x0040–0x004B	Reserved	
0x004C	IMA RAM Read Data register 4 (IMA_READ_DATA_4)	*Section 54.2.3.8*
0x0050	IMA RAM Read Data register 3 (IMA_READ_DATA_3)	*Section 54.2.3.8*
0x0054	IMA RAM Read Data register 2 (IMA_READ_DATA_2)	*Section 54.2.3.8*
0x0058	IMA RAM Read Data register 1 (IMA_READ_DATA_1)	*Section 54.2.3.8*
0x005C	IMA RAM Read Data register 0 (IMA_READ_DATA_0)	*Section 54.2.3.8*

54.2.3.1 IMA Control register (IMA_CTRL)

The IMA CTRL register controls read/write accesses. Since the IMA may interfere with proper read/write access of RAMs, a locking scheme is implemented to prevent accidental activation of the IMA. Using this scheme, the IMA read/write accesses can be locked when desired.

Offset: 0x00 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 947. IMA Control register (IMA_CTRL)

Table 682. IMA_CTRL field descriptions

Field	Description
0 READ	IMA Read This bit controls whether the IMA performs a read or write 0 The IMA access is a write. 1 The IMA access is a read.

54.2.3.2 IMA Enable Access register (IMA_ENABLE)

The Enable Access Register is a single-bit register that starts an IMA access. The bit automatically clears itself after two clocks thus providing an enable pulse.

Note: Back to back writes to the enable bit is not valid. It still generates a two-clock enable pulse and is considered as a single access initiation.

Offset: 0x04 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 948. IMA Enable Access register (IMA_ENABLE)

Table 683. IMA_ENABLE field descriptions

Field	Description
0 EN	IMA Enable This bit controls RAM access initiation by the IMA. Once set, this bit is cleared after two clocks to provide an enable pulse. 0 No IMA access occurs or the IMA has finished the previous access. 1 Start an IMA access or an IMA access is occurring.

54.2.3.3 IMA Status register (IMA_STATUS)

The IMA Status register shows the current status of the IMA as to which operations are unlocked.

Offset: 0x08 Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	WRITE_LOCK	0	0	0	0	0	0	0	READ_LOCK
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 949. IMA Status register (IMA_STATUS)

Table 684. IMA_STATUS field descriptions

Field	Description
8 WRITE_LOCK	The ability of the IMA to do write accesses. The WRITE_LOCK bit shows the current status of the lock on IMA write accesses. If set, the IMA does not write to the memories. If cleared, the IMA is allowed to do writes to the memories. 0 Write accesses from the IMA to memories are allowed. 1 Write accesses from the IMA to memories are not allowed. Any memory writes from the IMA are ignored.
0 READ_LOCK	The ability of the IMA to do read accesses. The READ_LOCK bit shows the current status of the lock on IMA read accesses. If set, the IMA does not do reads from the memories. If cleared, the IMA is allowed to do reads from the memories. 0 Read accesses from the IMA to memories are allowed. 1 Read accesses from the IMA to memories are not allowed. Any memory reads from the IMA are ignored.

54.2.3.4 IMA RAM Select register (IMA_SLCT)

The IMA RAM Select register provides a method of selecting which RAM memory block is accessed. This method is used because RAM is to be accessed in a non-functional way, such as accessing all bits including the ECC, and accessing on RAM word boundaries instead of byte or 32-bit word boundary.

Note: Only one RAM can be selected at a time. Each SoC has a different encoding for the RAMs. Refer to the Device Configuration chapter for details.

Offset: 0x0C Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	ROW_SLCT														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	ARRAY_SLCT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 950. IMA RAM SELECT register (IMA_SLCT)

Table 685. IMA_SLCT field descriptions

Field	Description
30:16 ROW_SLCT	The ROW_SLCT value specifies the row of the current RAM array under test to be selected. Hence the largest RAM which the IMA can access is 32 KB RAM.
5:0 ARRAY_SLCT	The ARRAY_SLCT specifies the RAM array to be tested. Up to 64 RAM instances can be supported.

54.2.3.5 IMA Write Unlock register (IMA_WRITE_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The WRITE_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for writes.

Offset: 0x10 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	WRITE_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WRITE_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 951. IMA Write Unlock register (IMA_WRITE_UNLOCK)

Table 686. IMA_WRITE_UNLOCK field descriptions

Field	Description
31:0 WRITE_KEY	Key to unlock the IMA. – 0x04A43F95 is the first key that has to be written to start the unlock process. – 0xE4A9EBF7 is the second key that has to be written to unlock the IMA.

54.2.3.6 IMA Read Unlock register (IMA_READ_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The READ_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for reads.

Offset: 0x14 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	READ_KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	READ_KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 952. IMA Read Unlock register (IMA_READ_UNLOCK)

Table 687. IMA_READ_UNLOCK field descriptions

Field	Description
31:0 READ_KEY	Key to unlock the IMA Read Access to RAMs. – 0xF06AB5BC is the first key that has to be written to start the unlock process. – 0x14081B56 is the second key that has to be written to unlock the IMA.

54.2.3.7 IMA RAM Write Data register n (IMA_WRITE_DATA_*n*)

The IMA_WRITE_DATA_*n* registers hold the data values to be written to the RAM. The number of active data bits depends on the number of bits for the RAM selected. Upper unused write data bits are ignored during the RAM access. With IMA_CTRL[READ] set to 0, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA begins a write access.

Offset: 0x2C (WRITE_DATA_4), 0x30 (WRITE_DATA_3), 0x34 (WRITE_DATA_2), 0x38 (WRITE_DATA_1), 0x3C (WRITE_DATA_0) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WRITE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WRITE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 953. IMA RAM Write Data register *n* (IMA_WRITE_DATA_*n*)

Table 688. IMA_WRITE_DATA_*n* field descriptions

Field	Description
31:0 WRITE	Contains data to be written to RAM. – WRITE_DATA_4[WRITE] contains write data bits[159:128] – WRITE_DATA_3[WRITE] contains write data bits[127:96] – WRITE_DATA_2[WRITE] contains write data bits[95:64] – WRITE_DATA_1[WRITE] contains write data bits[63:32] – WRITE_DATA_0[WRITE] contains write data bits[31:0]

54.2.3.8 IMA RAM Read Data register n (IMA_READ_DATA_*n*)

The IMA_READ_DATA_*n* registers hold the data values read from the RAM. The number of active data bits depends on the number of bits for the RAM selected. Upper unused READ DATA bits are updated to zeroes during the RAM access. With IMA_CTRL[READ] set to 1, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA begins a read access. Performing a read operation on IMA_READ_DATA_*n* registers causes data from RAM to be latched into these registers and presented to the CPU on the next clock.

Offset: 0x4C (READ_DATA_4), 0x50 (READ_DATA_3), 0x54 (READ_DATA_2), 0x58 (READ_DATA_1), 0x5C (READ_DATA_0) — Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	READ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	READ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 954. IMA RAM Read Data register n (IMA_READ_DATA_*n*)

Table 689. IMA_READ_DATA_*n* field descriptions

Field	Description
31:0 READ	Contains data to be written to RAM. – READ_DATA_4[READ] contains read data bits[159:128] – READ_DATA_3[READ] contains read data bits[127:96] – READ_DATA_2[READ] contains read data bits[95:64] – READ_DATA_1[READ] contains read data bits[63:32] – READ_DATA_0[READ] contains read data bits[31:0]

54.2.4 IMA module functional description

This section describes the operation of the IMA, beginning with the unlocking of reads and writes, the software initialization, then the software interface for generating single-bit ECC errors, generating multiple-bit ECC errors, and a software supported memory BIST.

54.2.4.1 Initialization sequence

This section describes which registers are reset due to hardware reset and what locations the user must initialize prior to doing an access by the IMA.

54.2.4.2 Hardware controlled initialization

In the IMA, registers and control logic are reset by hardware (system reset). A system reset deasserts output signals and resets general configuration bits.

54.2.4.3 User initialization (prior to asserting IMA_ENABLE[EN])

The user needs to initialize portions of the SoC prior to setting the IMA_ENABLE[EN] bit. The exact values depend on the particular application. Refer to the specific chapter's memory map to see how to turn off that peripheral so that accesses are not happening at the same time as the IMA. Without the peripheral shut down, there is a small chance that the IMA and the peripheral can interfere with each other in accessing the RAM.

54.2.4.3.1 Shut down the peripheral

The IMA can cause problems with the system or the peripheral if both the IMA and the other system components are accessing the RAM during normal operation.

To shut down the peripheral, read the peripheral's documentation. The documentation describes how to put the peripheral into a quiet state so that it does not access the RAM during the IMA access. If the other functional path is running at the same time, the data for the other peripheral are corrupted and unknown.

54.2.4.3.2 Unlocking reads

The IMA does not allow a read to any memory unless it has been unlocked. This is a safety feature so that it takes two unique writes to the READ_KEY register to unlock writes. To unlock the read feature, the system has to do the following:

1. Write 0xF06AB5BC to READ_KEY.
2. Write 0x14081B56 to READ_KEY.

The IMA has a feature to make the READ_KEY lock in the case where an incorrect key is written to READ_KEY register.

54.2.4.3.3 Unlocking writes

The IMA does not allow a write to any memory unless it has been unlocked. This is a safety feature so that it takes two unique writes to the WRITE_KEY register to unlock writes. To unlock the write feature, the system has to do the following:

1. Write 0x04A43F95 to WRITE_KEY.
2. Write 0xE4A9EBF7 to WRITE_KEY.

The IMA has a feature to make the WRITE_KEY lock in the case where an incorrect key is written to WRITE_KEY register.

54.2.4.4 IMA lock

The IMA module also allows for read/write access locking via an input pin. If this input is asserted, both read/write accesses via IMA are locked. This locking scheme overrides the above KEY based locking scheme. Hence it provides additional safety over the KEY based locking method. However, usage of this feature is entirely SOC specific.

54.2.4.5 IMA access

There are two differences between a read access and a write access. The first difference is that the IMA_STATUS[WRITE_LOCK] bit has to be cleared for writes and the IMA_STATUS[READ_LOCK] has to be cleared for reads. The second difference is that the IMA_CTRL[READ] has to be set for reads and IMA_CTRL[READ] has to be cleared for writes.

The following sections show the different accesses that are available depending on the options.

54.2.4.5.1 Read access

Read Accesses are expected to follow the sequence below:

1. Unlock Read Access by performing appropriate consecutive writes to READ_UNLOCK register.
2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Set the READ bit in the IMA_CTRL register.
4. Set the EN bit in the IMA_ENABLE register to initiate read access. The EN bit self-clears after two clocks.
5. Read data from READ_DATA registers. This causes data from RAM to be latched into the IMA and returned to the CPU on the next clock.

54.2.4.5.2 Write access

Write Accesses are expected to follow the sequence below:

1. Unlock Write Access by performing appropriate consecutive writes to WRITE_UNLOCK register.
2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Write the Write Data in WRITE_DATA registers.
4. Clear the READ bit in the IMA_CTRL register
5. Set the EN bit in the IMA_ENABLE register to initiate write access. The EN bit self-clears after two clocks.

54.2.5 Application information

Caution: The IMA inhibits correct operation of the underlying peripherals. The application software is responsible for ensuring there is not a resource conflict.

The primary function of the IMA module is to provide a means for the system to program invalid ECC bits and see that the appropriate peripheral is able to detect the error. For a single-bit error, the peripheral must be able to correct the error. For a multiple-bit error, the peripheral must report an error.

55 Self-test control unit (STCU3)

55.1 Introduction

55.1.1 Overview

The self-test control unit (STCU3) is a comprehensive programmable hardware module that controls the self-test sequence applied during the offline conditions.

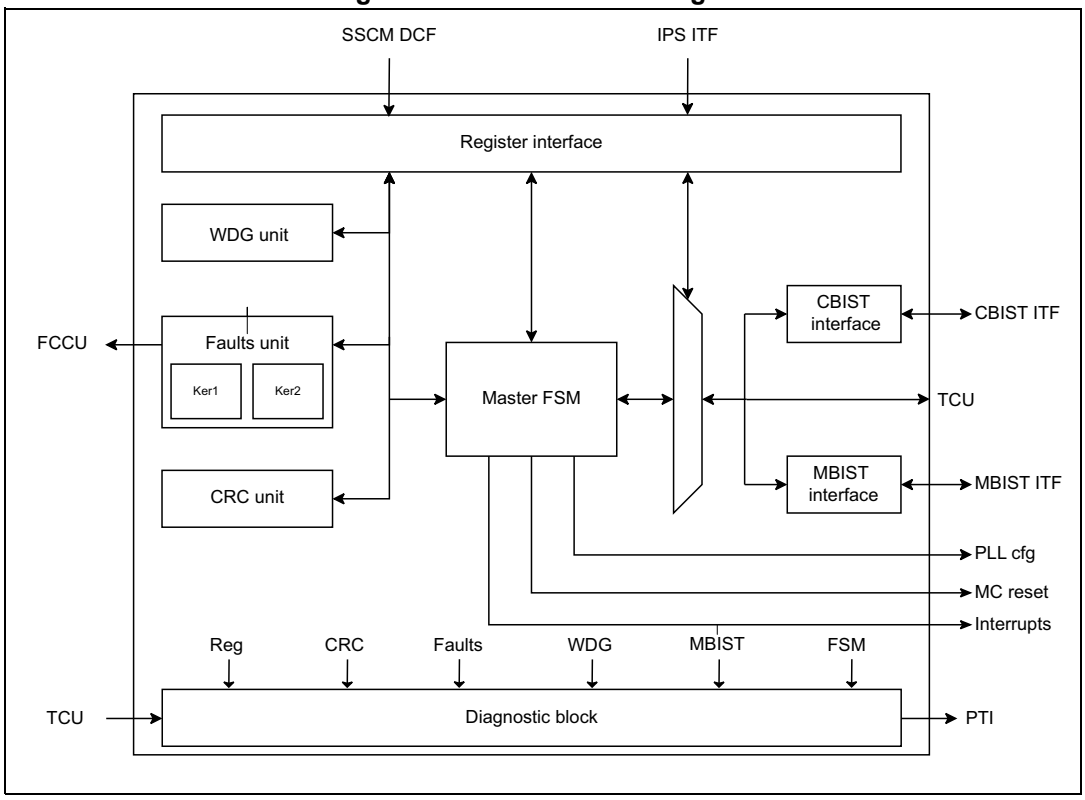
The hardware can manage different built-in self test (BIST) types of the device.

The STCU includes the DCF bus to load the self-test parameters (such as BIST scheduling activity, unrecoverable faults or recoverable faults management, CRC and MISR expected values, PLL management) from NVM memory during the offline self-test phase. This interface is only able to write the configuration parameters and start the self-test execution once the STCU3 global reset has been applied. Register access by software is granted by an IPS interface to check the results of the offline self-test.

To increase the security of this module, a different security key sequence is also required during offline self-test to unlock write access to the STCU3 registers.

55.1.2 Block diagram

Figure 955. STCU3 block diagram



The STCU3 module includes the following submodules:

- **Register interface**

  The register includes the register, the security key logic, IPS and SSCM DCF interface.
- **CRC**

  This module constitutes self-checking architecture of the STCU3. It samples set of selected MBIST signals when STCU3 is running.
- **Diagnostic block**

  This module constitutes core of debug to be used in case of internal trouble. The critical and useful signals of the STCU3 module can be observed depending on the debug control signals, on external pads.
- **Fault interface**

  This module collects the error conditions respective to MBIST and CBIST execution, and STCU3 internal failures and depending on UF/ RF configuration of each of them, sets the global STCU3 UF/ RF flag. It also manages the UF/ RF fault lines, the mask control of EOUT signals towards FCCU, and set/clear injection mechanism provided by the FCCU. To improve the intrinsic safety of this critical logic, the generation logic is duplicated.
- **Clock generation block**

  This module manages the internal and the MBIST TCK clock pre-scaler, the internal clock gating power saving and the wake-up clock feature.
- **Run watchdog FSM**

  A dual function logic:
  - After a reset event initializes STCU3, it is used as a parameterized watchdog timeout.
  - During MBIST and CBIST runs, it is used as a programmable watchdog timer to check whether the MBIST and/or CBIST have/has completed in the designated time slot.
- **Auto-lock watchdog block**

  A parameterized watchdog used to auto-lock the STCU3 access, forcing a reset condition on the double security key registers.
- **STCU3 master FSM block**

  This module includes the main FSM of the STCU3 used to coordinate and schedule all the operations performed during the self-test sequence.
- **MBIST FSM block**

  This module includes the shifter register and the respective state machine used to program the MBIST registers and read back the data to be checked at the end of each self-test operation.
- **MBIST interface**

  This module includes the interface between the MBIST controller and TCU (debug/test feature) or STCU3 (user application) controller of the device.
- **CBIST FSM block**

  This module includes the respective state machine used to trigger CBIST collector modules and read back the data to be checked at the end of each self-test operation.
- **CBIST interface**

  This module includes the interface between the CBIST collector modules and the STCU3 (user application) controller of the device.

55.2 Features

- Interface features:
  - Dual Register Interface: SSCM DCF and IPS
  - DCF write once register interface
  - IPS read and write (depending on SSCM enable signals and STCU_CFG.WRP bit) register interface
  - Double security key protection mechanism
  - Hardware and Software self-test bypass capability
  - Programmable PLL direct control during offline mode
  - Optional PLL Lock monitoring during offline mode
- Self-test execution features:
  - Programmable internal clock pre-scalar to reduce internal and MBIST TCK clock
  - Hardware DCF write masking capability
  - Programmable scheduler for MBIST and CBIST execution
  - STCU3 internal errors offline programmable failure severity (UF/RF)
  - STCU3 internal errors offline status flags
  - STCU3 ITF/WDG clock wake-up mechanism when software application writes double security key sequence (available when STCU_CFG[WRP] bit is active)
  - STCU3 ITF/WDG auto power-saving when WDG timeout is detected after software application writes double security key sequence (available when STCU_CFG[WRP] bit is active)
- Offline mode support:
  - Self-Test execution capability during power up sequence
- Watchdog support:
  - Flags timeout in case of offline initialization errors
  - Timeout in case of Write Access Auto-lock after double key sequence
  - Timeout in case Self-Test sequence does not complete in specified interval specified by Programmable WDG timer value written into WDG Register
- Faults interface support:
  - Error lines for FCCU signaling (UF/RF)
  - Redundant fault trigger generation logic for safety
  - Fault trigger injection mechanism
- CBIST support:
  - Up to 16 CBIST controller management
  - CBIST Sequential / Concurrent execution
  - CBIST offline status flags
  - CBIST offline programmable failure severity
- MBIST support:
  - Up to 16 MBIST controller management
  - Up to 1024 memory cuts support (which can be distributed amongst multiple MBIST controllers or single MBIST controller)
  - MBIST offline status flags
  - MBIST Sequential / Concurrent execution

  - Programmable MBIST delayed concurrent run
  - MBIST offline programmable failure severity
- Auto power saving:
  - Self-Test is complete
  - WDG Timeout detected
  - Bypass mode selected
- CRC check feature:
  - 32-bit CRC for STCU3 self-checking capability (enabled by STCU_CFG[CRCEN] bit)
  - CRCE register access to perform redundant check by software
- Diagnostic features:
  - Programmable diagnostic interface
- IPS bus interface
  The IPS bus is a slave bus used for configuration purposes via CPU core. The following bus write operations (contiguous byte enables) are supported.
  - Word (32-bit) data read operations to any registers
  - Low and high half-words (16 bits, data [31:16] or data [15:0]) data read operations to any registers
  - Byte (8 bits, data [31:24] or data [23:16] or data [15:8] or data [7:0]) data read operations to select registers (refer to register description)
  - Any other operation write (free byte enables or other operations) must be avoided, else transfer error is generated
  - Only 32-bit read operation allowed via IPS interface
  - The self-test sequence parameters are loaded before offline self-test is executed
  - The IPS access on some specific registers is allowed by software only when the bit STCU_CFG[WRP] is cleared

55.3 Functional description

55.3.1 FSM description

STCU3 has 4 state machines that work together:

- Master FSM: The core unit of STCU3 module; it coordinates all the self-test operations and other state machines.
- MBIST state machine: used to program the MBIST parameters and to retrieve the respective results depending on the parameters stored in the STCU3 registers and under the control of the master state machine.
- CBIST state machine: used to trigger the CBIST collector modules for self-test of RCCU structures spread across device, and to retrieve the respective results in STCU3 registers and under the control of the master state machine.
- Watchdog state machine: used to evaluate all the schedule time for MBIST and CBIST and the timeouts in case of incorrect STCU3 programming.

55.3.2 DCF interface

The DCF interface is used to program the configuration parameters of STCU without CPU intervention after a reset trigger event initializes it.

55.3.3 Reset management

The asynchronous reset line initializes the STCU module status, forcing the offline self-test condition and the cleaning up of all the registers and state machines of the module.

The reset signals generated by the STCU3 module at the end of the self-test execution phase depends on the self-test conditions reported in the subsequent subsections.

55.3.3.1 Offline reset generation

At the end of the offline self-test execution, the STCU raises the global functional reset. In case of unrecoverable fault during offline self-test execution, the STCU generates a request of destructive reset for the whole device.

55.3.4 BIST scheduling

The STCU3 module is designed to be very flexible, allowing the programming of the parallel or serial execution of the MBIST or CBIST depending on the power, timing and coverage constraints.

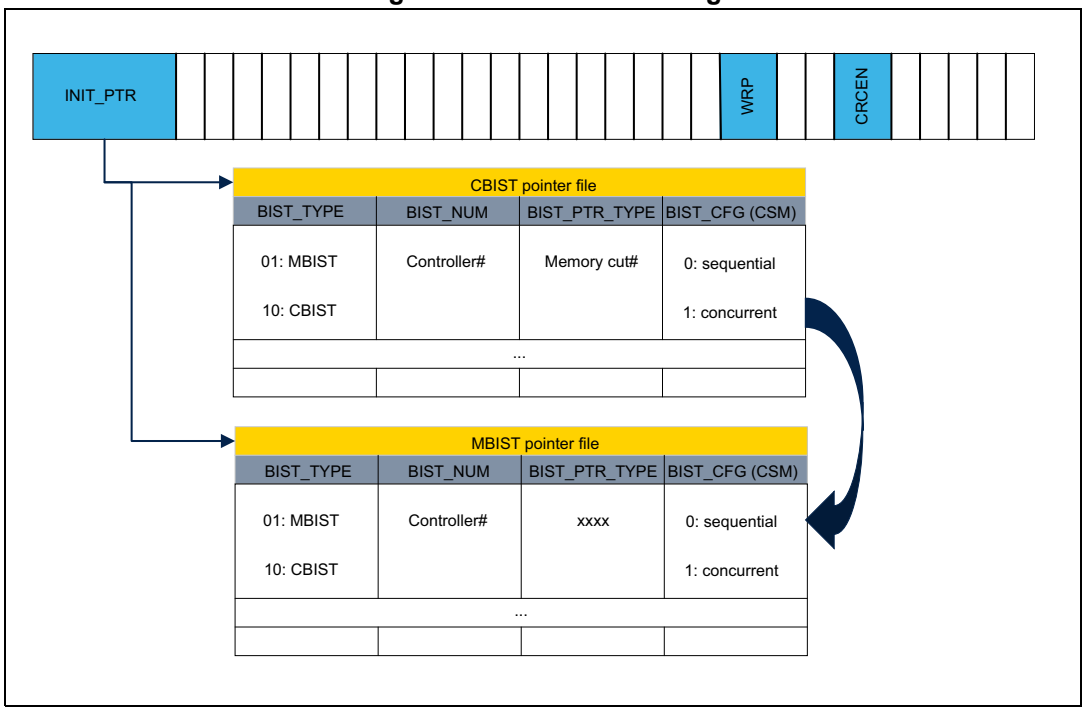
The mechanism used to provide this flexibility is a set of separate register file for each BIST operation, where the starting pointer is the bit field STCU_CFG[INIT_PTR]. The initial pointer field selects BIST type to be executed first. The register file for each BIST type, indicates the BIST execution parameters required for self-test sequence.

In case of MBIST, for each register in the MBIST register file, the bit field STCU_MBISTn_PTR.BIST_TYPE indicates the next BIST type scheduled for execution. The bit field STCU_MBISTn_PTR.BIST_CTRL_NUM indicates the ID of controller to be programmed for execution, while the memory cut to be tested is indicated by the bit field STCU_MBISTn_PTR.BIST_PTR_VAL. The bit field STCU_MBISTn_PTR.BIST_CFG indicates whether the next BIST test is executed parallelly/sequentially with the currently selected BIST (concurrent/sequential scheduling is meant for same BIST type).

In case of CBIST, for each register in the CBIST register file, the bit field STCU_CBISTn_PTR.BIST_TYPE indicates the next BIST type scheduled for execution. The bit field STCU_CBISTn_PTR.BIST_CTRL_NUM defines the id of the controller to be programmed for execution. The bit field STCU_CBISTn_PTR.BIST_CFG indicates whether the next BIST test is executed parallelly/sequentially with the currently selected BIST.

The order of execution must start with CBIST, then MBIST.

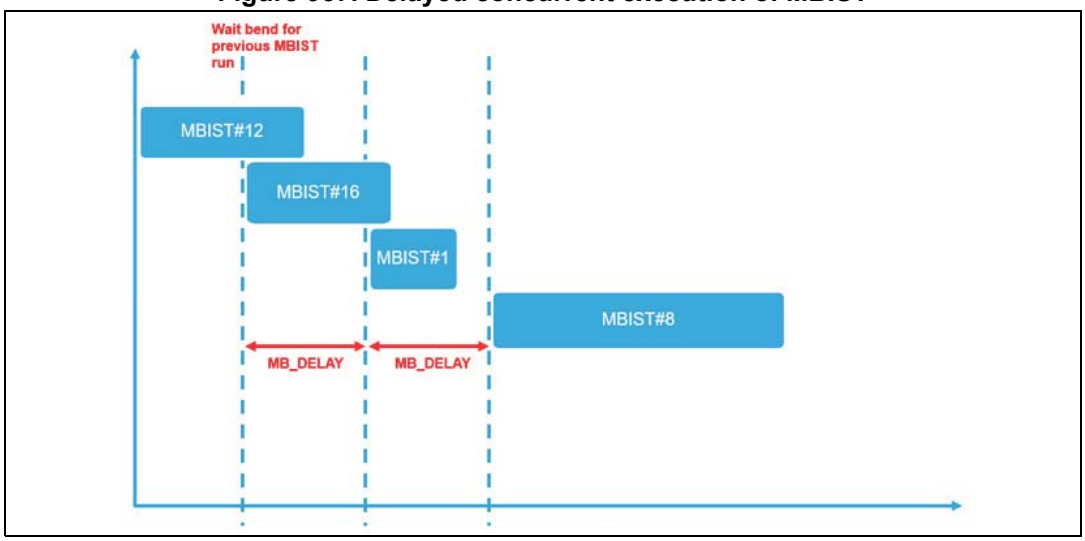
Figure 956. BIST scheduling



Delayed concurrent execution of MBIST is supported with the purpose of smoothing the power consumption transient. The next BIST execution is delayed by the value indicated in STCU_RUN_DELAY register fields (STCU_RUN_DELAY.MB_DELAY for MBIST delayed execution).

The figure depicts execution of delayed concurrent MBIST test for controller #16, 1 and 8. The MBIST delayed concurrent test is executed post sequential execution of MBIST controller #12 (arbitrary memory cut test is assumed).

Figure 957. Delayed concurrent execution of MBIST



55.3.5 PLL interface

The STCU module is able to directly control all the PLLs during the offline self-test operations depending on the status of the STCU_RUN[MBPLLEN] and STCU_RUN[CBPLLEN] as described in Register description section.

When STCU_RUN[RUN] is set to start the offline self-test operations, the STCU3 takes control of the PLL and the STCU3 changes the PLL clock configuration parameters according to the values programmed in the STCU_PLLn_CFG register (for each PLL selected for programming).

When the STCU_RUN[MBPLLEN] is active and the MBIST is currently selected, the STCU waits for the PLL lock looking at the pll_lock[n] signal. As soon as the lock signal of all selected PLL goes high, the STCU forces the clock to switch to the PLL output clock signal, allowing the MBIST to proceed at the PLL output frequency. At the end of the MBIST run, the clock source reverts to the original one.

Same behavior applies when the STCU_RUN[CBPLLEN] is active and CBIST is currently selected.

55.3.6 Interrupt interface

STCU3 has 3 interrupt sources:

- Triggered by MBIST:
  - Enabled by programming STCU_INT_CTRL[MB_INT_EN]
- Triggered by CBIST:
  - Enabled by programming STCU_INT_CTRL[CB_INT_EN]

The INT_MB_STATUS bit in the STCU3 Interrupt Flag Register (STCU_INT_STATUS) flags interrupt condition and allows the clearing of interrupt requests. The interrupt bit is cleared by writing the respective bit to 1 to enable the next interrupt request.

55.3.7 Fault collector interface

The Fault collector interface is the hardware flag mechanism to the system, indicating the occurrence of an unrecoverable failure (UF) or a recoverable failure (RF) during the self-test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. The Fault collector interface allows the user application to check the integrity of the UF and RF connection lines between the STCU3 and the fault collector. Refer to the description of fault injection mechanism to understand how the UF/RF set and clear mechanism works.

55.3.8 Watchdog support

Key unlocking and monitoring of BIST is the main operation of both the watchdogs. The bypass mechanism is another feature of watchdogs. All these operations are being handled by the 2 watchdogs mentioned (RUN/ AUTOLOCK) which perform the following operations:

- Program and run the self-test sequence (safety mode).
- Bypass the self-test sequence by setting BYP bit in STCU_RUN register.

55.3.8.1 During initialization (configuration loading)

Two watchdog features are implemented during initialization:

- If there are faults during the initialization phase which prevent the selection of one of the two operating modes mentioned in *Section 55.3.8* (safety mode and bypass mode), a watchdog timeout flags the incorrect behavior.
- There is a hardware watchdog that, after 1024 system clock cycles, locks the write access of registers and requires reinsertion of the STCU_SKC keys. This feature is particularly useful if STCU_CFG[WRP] = 0 during the software self-test configuration as the software application may enable write access to the STCU registers.

55.3.8.2 During self-test execution

The execution time of all the BISTs present is configured in STCU_WDG register to account for overall execution time of self-test sequence. If the BIST execution does not complete in the time (calculated as STCU_WDG_VAL [31:0] x system clock cycles), execution is interrupted (meaning all including current and upcoming BIST runs are skipped) and a failure is flagged in the corresponding registers.

55.3.9 CRC support

The CRC block is used to increase the intrinsic coverage of the STCU3 module and implements a 32-bit ethernet protocol polynomial to evaluate the signature of the most important and critical signals of MBIST operations (control signals/ status signals).

Note: STCU3 state machine is not covered by CRC.

This feature is switched off for default, but it can be turned on by setting the STCU_CFG[CRCEN] bit. Once this bit is set, the expected signature must be programmed in the STCU_CRCE register. At the end of the self-test execution, the STCU updates the content of the STCU_CRCR register with the evaluated signature and compares this value with the expected one previously programmed in STCU_CRCE. In case of mismatches, an error is flagged depending on the fault mapping selected in the STCU_ERR_FM register. CRC support is implemented for MBIST, but not for CBIST.

55.4 Programming guidelines

The STCU3 module is designed to give high flexibility during the offline self-test. This section shows the programming operations that have to be performed to correctly use this module in two typical use cases:

- Offline self-test sequence after a reset trigger event
- Bypass mode of the self-test sequence after a reset trigger event

55.4.1 Offline self-test sequence

This is the typical mode of using the STCU3 module after reset trigger event is applied to STCU3. The DCF bus is used to program the STCU3 schedule and MBIST and CBIST execution parameters stored in the NVM memory. The target is to cover the amount of

physical defects in the digital logic and in the system RAM or ROM according to the system specifications. The sequence is the following:

- Unlock the offline STCU3 access writing the Key1/Key2 sequence in the STCU_SKC register.
- If PLL usage is enabled, program the STCU_PLL_CFG value to set the required system frequency, PLL selection and RCC input selection.
- Program the STCU_PLL_MISC register, for PLL configuration as per SoC requirement.
- Program the STCU_CFG register to define
  - the core and MBIST TCK clock pre-scaling factor setting the CFG_CLK bits.
  - the CRCREN bit to enable the self-checking feature.
  - set the initial pointer to indicate the first BIST type to be executed (execution starts from the top of register file).
- If the internal CRC comparison has been enabled, program the CRCE value expected at the end of the offline self-test sequence in the STCU_CRCE register.
- Program the STCU_WDG register to define the time budget assigned for MBIST and CBIST execution.
- Program the MB_DELAY field to delay the MBIST start when many MBIST are run concurrently to mitigate the potential transient power issue during start-up phase.
- Program the STCU3 internal fault reaction conditions (UF/RF) setting the STCU_ERR_FM register.
- Program the MBIST fault reaction conditions (UF/RF) setting the STCU_MBn_UFMm, depending on the number of memories per controller. The number/ width of registers for MBIST fault mapping is dependent on the number of memory cuts implemented per MBIST controller.
- Program the CBIST fault reaction conditions (UF/RF) setting the STCU_CB_UFM register.
- Program the STCU_MBn_TM register, for specifying PMOSEN or MBU bit field to define the algorithm to be used during MBIST run.
- Program the STCU_MBISTn_PTR and STCU_CBISTn_PTR to define the BIST execution sequence.
- Offline self-test starts when the sys_slftst_en is high depending on PLL management and MBIST or CBIST execution order when pll_lock is high.
- Wait the WDGEOC execution run time specified in the STCU_WDG register.
- The STCU3 switches off the core clock at the end of the self-test run and releases the PLL control if this feature is enabled.
- The STCU3 forces the stcu_rst_func_b of being active and wait that rst_phase1_b signal from RCC to validate the reset pulse. Only at this stage, the signal stcu_rst_func_b is released allowing the reboot of the system in USER mode. If there are failures, the STCU3 flags these failures toward the FCCU rising one/both the stcu_ncf and stcu_cf lines.
- The STCU3 resets the STCU_RUN[RUN] bit and the device exits the boot sequence. If there are failures, the STCU3 flags these failures to the fault collector by raising one or both stcu_ncf and stcu_cf lines.
- The software application reads the STCU_MBn_STATUSm flag registers, depending on the number of memories per controller (scheduled for execution during MBIST test), to check the failed RAM or ROM memory BIST when UF or RF is detected.

- The software application reads the STCU_MBn_ENDFLAGm flag registers, depending on the number of memories, to check the RAM or ROM memory BIST still running when UF or RF is detected.
- The software application reads the bit INV, ENGE, CRCS, WDTO, LOCKE (for each PLL enabled during self-test execution) of the STCU_ERR_STAT register to check whether there has been an internal STCU3 engine/parameters failure when UF or RF is detected.
- If the CRCEN bit is enabled, the software application reads the CRCE and CRCR registers to check the coherence with the bit CRCS of the STCU_ERR_STATUS register.

55.4.2 Bypass user mode

In this mode, the user application parameters stored in NVM<var> and read by SSCM are written to skip the self-test procedure after a reset trigger event is applied. It might be useful if the device is not configured for Safety applications. Follow this sequence:

1. Unlock the STCU3 access writing the offline Key1/Key2 sequence in the STCU_SKC register.
2. Set the BYP bit in the STCU_RUN register.
   After the BYP bit is set, the core clock is switched off, the self-test sequence is not applied and the system can wake up and start the user application.

55.5 Memory map and register description

55.5.1 Memory map

Table 690. STCU3 memory map

Offset (Hex)	Registers
Configuration registers	
0x0000	*STCU offline RUN register* (*STCU_RUN*)
0x0004—0x0007	Reserved
0x0008	*STCU security key register* (*STCU_SKC*)
0x000C—0x000F	Reserved
0x0010	*STCU configuration register* (*STCU_CFG*)
0x0014	*STCU delay value register* (*STCU_RUN_DELAY*)
0x0018	*STCU PLL0 configuration register* (*STCU_PLL0_CFG*)
0x001C—0x003B	Reserved
0x003C	*STCU end of count register* (*STCU_WDG*)
0x0040—0x0044	Reserved
0x0048	*CRC expected signature register* (*STCU_CRCE*)
0x004C	*CRC read signature register* (*STCU_CRCR*)
0x0050	*STCU error status register* (*STCU_ERR_STATUS*)
0x0054	*STCU error fault mapping register* (*STCU_ERR_FM*)

Table 690. STCU3 memory map (continued)

Offset (Hex)	Registers
0x0058—0x0603	Reserved
MBIST configuration register	
0x0604	*STCU MBIST0 test mode selection register* (*STCU_MB0_TM*)
MBIST status registers	
0x0644	*STCU offline MBIST0 status register* (*STCU_MB0_STATUS0*)
0x0648	*STCU offline MBIST0 status1 register* (*STCU_MB0_STATUS1*)
0x064C—0x06C3	Reserved
0x06C4	*STCU offline MBIST0 endflag register* (*STCU_MB0_ENDFLAG0*)
0x06C8	*STCU offline MBIST0 endflag1 register* (*STCU_MB0_ENDFLAG1*)
0x06CC—0x0843	Reserved
0x0844	*STCU MBIST0 Unrecoverable FM register* (*STCU_MB0_UFM0*)
0x0848	*STCU MBIST0 Unrecoverable FM register* (*STCU_MB0_UFM1*)
0x084C—0x2F7F	Reserved
CBIST status registers	
0x2F80	*Offline STCU CBIST status register* (*STCU_CB_STATUS*)
0x2F84	*Offline STCU CBIST endflag register* (*STCU_CB_ENDFLAG*)
0x2F88—0x2F8F	Reserved
0x2F90	*STCU CBIST unrecoverable fault mapping register* (*STCU_CB_UFM*)
0x2F94—0x303F	Reserved
MBIST pointer registers (n=0 to 1023)	
0x3040 + n*0x4	*STCU MBISTn pointer register* (*STCU_MBISTn_PTR*)
0x4040—0x407F	Reserved
CBIST pointer registers (n = 0 to 15)	
0x4080 + n*0x4	*STCU CBISTn pointer register* (*STCU_CBISTn_PTR*)

55.5.2 Register description

55.5.2.1 Configuration register description

STCU_RUN **STCU offline RUN register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				RUN_BYP	MBPLLEN	RESERVED	CBPLLEN	RESERVED							RUN
R																				RW	RW	R	RW	R							RW

Address: STCU3BaseAddress + 0x0000

Type: RW

Reset: 0x00000000

Description: STCU Hardware RUN configuration register to start offline self-test procedure.

[11] **RUN_BYP**
0: Offline self-test is active. If the STCU_RUN[RUN] bit is not set before the hardcoded WDG timeout, the WDTO error is generated in the STCU_ERR_STAT register.
1: Offline self-test is bypassed and access to STCU2 is locked until the STCU_SCK register is written according to the request access mode.

[10] **MBPLLEN**
0: Offline MBIST is executed on default system clock (RCOSC).
.
1: Offline MBIST is executed enabling the on-chip PLL control interface selecting the parameters defined in STCU_PLL_CFG register. System Clock Mux is directly controlled by the STCU via STCU_PLLn_CFG[SYSCLK_SEL] register.

[8] **CBPLLEN**
0: Offline CBIST is executed without using the on-chip PLL.
1: Offline CBIST is executed enabling the on-chip PLL control interface selecting the parameters defined in STCU_PLL_CFG register.

[0] **RUN**: RUN bit is automatically cleared by STCU when offline self-test procedure has been completed.
0: Idle.
1: Offline self-test procedure is running.

STCU_SKC STCU security key register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SKC																															
W																															

Address: STCU3BaseAddress + 0x0008

Type: W

Reset: 0x00000000

Description: STCU double security key code mechanism needed for write access to other STCU registers. Depending on the offline test step, the two keys are different. Byte write operation is not allowed as the full key has to be recognized as one unit.

In case of invalid access or sequence transfer error on the IPS or SSCM bus is asserted depending on the selected source, and STCU write access is locked until the correct sequence is applied.

If the STCU register access lasts more cycles than defined in the hard-coded WDG timeout, STCU write access is locked and the WDG and register ITF clocks are switched off.

To extend the STCU register access cycles before the hard-coded WDG timeout expires, only Key2 needs to be applied.

[31:0] **SKC**: STCU2 security key code for offline test
0xD3FEA98B = Key1 to unlock the write access the STCU.
0x2C015674 = Key2 to unlock the write access the STCU.

STCU_CFG STCU configuration register

31 30 29 28	27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9	8	7 6	5	4 3	2 1 0
RESERVED	INIT_PTR	RESERVED	WRP	RESERVED	CRCEN	RESERVED	CFG_CLK
R	RW	R	RW	R	RW	R	RW

Address: STCU3BaseAddress + 0x0010

Type: RW

Reset: 0x00000000

Description: STCU offline global configuration. Access depends on state of WRP bit as follows:

– If software operations write to other register bits, a transfer error is raised if the value of the selected byte that is written differs from the current status of the register.

[27:24] **INIT_PTR**: INIT_PTR defines the logical pointer to the first MBIST to be scheduled when the self-test procedure is enabled. It indicates the first pointer of selected BIST type.
If both CBIST and MBIST are planned, the INIT_PTR must be set to 0x2 (CBIST first).
0x1: MBIST is selected.
0x2: CBIST is selected.

[8] **WRP**
0: Specific STCU registers can be written through IPS bus interface after offline self-test sequence has completed.
1: Specific STCU registers cannot be written though IPS even after offline self-test sequence has completed, thus preventing any user application write operation.

[5] **CRCEN**
0: The CRC comparison is not performed and the STCU_ERR_STAT[CRCS][SW] and STCU_ERR_STAT[UF/RF] global flags are not updated in case of mismatches between STCU[CRCE] and STCU[CRCR] values.
1: The CRC comparison is performed and the status is updated in the respective flags of the STCU_ERR_STAT register.

[2:0] **CFG_CLK**: CFG_CLK defines the ratio between the STCU3 input clock with respect to the clock used to program MBIST, as well as STCU3 internal clock. The allowed configurations are:
000 PCLK1
001 PCLK1/2
010 PCLK1/3
011 PCLK1/4
100 PCLK1/5
101 PCLK1/6
110 PCLK1/7
111 PCLK1/8

STCU_RUN_DELAY STCU delay value register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MB_DELAY																RESERVED															
RW																R															

Address: STCU3BaseAddress + 0x0014

Type: RW

Reset: 0x00000000

Description: STCU concurrent BIST execution with delay value.

[31:16] **MB_DELAY**: MB_DELAY defines the delay between the MBIST starts when more than a single MBIST is selected to be executed concurrently with the purpose of smoothing the power consumption transient. The next MBIST execution is delayed by the MB_DELAY value after completion of first MBIST test. MBIST delay has to be used only when there is a change in MBIST controller.

0x00: No delay
0x01: 1 x 16 STCU2 Core clock cycles
0x02: 2 x 16 STCU2 Core clock cycles
....

STCU_PLL0_CFG STCU PLL0 configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL_EN	RESERVED	ODF2						RESERVED		ODF1						RESERVED	SYSCLK_SEL				IDF			RESERVED	LDF						
RW	R	RW						RW		RW						R	RW				RW			R	RW						

Address: STCU3BaseAddress + 0x0018

Type: RW

Reset: 0x00000000

Description: STCU offline PLL configuration to program the PLL0 only when offline MBIST/CBIST are executed and STCURUN[MBPLLEN/CBPLLEN] are set.

The R/W fields in this register are readable at any time. However, these fields can be written into only when offline self-test phase is active.

[31] **PLL_EN**: PLL enable

[29:24] **ODF2**: The value of this field drives the 2nd output division factor of the PLL embedded in the device

[21:16] **ODF1**: The value of this field drives the 1st output division factor of the PLL embedded in the device

[14:11] **SYSCLK_SEL**: Input clock selection for System Clock MUX when STCU_RUN[MBPLLEN]=1
XX00: RCOSC
XX01: reserved
XX10: PLL0
XX11: reserved

[10:8] **IDF**: The value of this field drives the input division factor of the PLL embedded in the device

[6:0] **LDF**: The loop division factor of the PLL embedded in the device

STCU_WDG STCU end of count register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDG_EOC																															
RW																															

Address: STCU3BaseAddress + 0x003C

Type: RW

Reset: 0x0004FFFF

Description: STCU end of count value to define timing budget for offline self-test execution providing protection against deadlocks. When the offline self-test sequence is run, it defines the timeout of the execution run.

When the offline self-test sequence is not run and the STCU_RUN[BYP] bit is not set, the bits 31 to 0 define the timeout before the STCU_ERR_STATUS[WDTO] bit is set and STCU core clock is switched off.

When the offline self-test sequence is run, it defines the timeout of the execution run. The R/W fields in this register are readable at any time. These fields can be written into when either of the following conditions is true:

– Offline self-test phase is active

[31:0] **WDG_EOC**: End of count value to define timing budget for offline self-test execution and check that everything is working correctly in given timeslot.
Allowed time delays:
0x00000000 1 x 16 STCU2 core clock cycles
0x00000001 2 x 16 STCU2 core clock cycles
....

STCU_CRCE CRC expected signature register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCE																															
RW																															

Address: STCU3_Address + 0x0048

Type: RW

Reset: 0x00000000

Description: CRC expected signature value of the CRC-32 (ethernet protocol). The CRC-32 bit engine is initialized at STCU_CRCE_RES and computes the CRC signature of the STCU critical signals. When STCU_CFG[CRCEN] = 1, it provides the self check capability of the STCU managing the CRCS[SW] bits of the STCU_ERR_STATUS register.

The R/W fields in this register are readable at any time. However, these fields can be written into when the following condition is true:

– Offline self-test phase is active.

[31:0] **CRCE**: CRC expected signature value

STCU_CRCR CRC read signature register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCR																															
RW																															

Address: STCU3BaseAddress + 0x004C

Type: RW

Reset: 0x00000000

Description: The STCU_CRCR register reports the value obtained at the end of the offline self-test sequence.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

[31:0] **CRCR**: Read CRC signature value

STCU_ERR_STATUS STCU error status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		UFSF	RFSF	RESERVED																LOCKE								WDTO	CRCS	ENGE	INVP
R		RW	RW	R																R								R	R	R	R

Address: STCU3BaseAddress + 0x0050

Type: RW

Reset: 0x00000000

Description: STCU Error status includes the status flags respective to the STCU internal error conditions that occurred during configuration or the offline self-test execution.

Access to this register is as follows:

– If the byte write capability is selected to write only the UFSF and RFSF, then there is no restriction in writing these bits.

– If the software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value being written to the other bits differs from their value currently stored in the register.

[29] **UFSF**:

0: No errors that trigger the unrecoverable faults condition.

1: There are errors that trigger the unrecoverable faults condition.

[28] **RFSF**:

0: No errors that trigger the recoverable faults condition.

1: There are errors that trigger the recoverable faults condition.

[11:4] **LOCKE**:

0: The PLL is correctly locked during the self-test sequence.

1: When the PLL is enabled, this flag highlights an unexpected PLL unlock event during the offline self-test sequence execution. The offline self-test run is stopped and the status of the current running MBISTs/CBISTs is saved in the respective registers.

[3] **WDTO**:

0: MBIST/CBIST time slot have been completed within the assigned watchdog time.

1: MBIST/CBIST time slot haven‘t been completed within the assigned watchdog time or there are internal mismatches among end of execution signals.

The conditions that flag failure are the following:

– The STCU_RUN[RUN] bit or the STCU_RUN[BYP] bit is not set before the watchdog reaches the end of count.

– Serial global BEND status or the direct BEND flag the MBIST run is not finished.

– Serial MBIST BEND status flags that at least one of the selected MBIST run is not finished.

– rccu_collector_done of the selected CBIST flags the CBIST run is/are not finished.

[2] **CRCS**: This flag is activated only when offline self-test is performed STCU_CFG[CRCEN] is set to 1, otherwise it is always forced to 0. This prevents the generation of the respective STCU_ERR_STAT[UFSF/RFSF] bits. In both the cases, the content of the STCU_CRCR register reports the current evaluated CRC.

0: Successful CRC comparison or comparison has been masked (STCU_CFG[CRCEN] programmed to 0).

1: Failed CRC comparison.

[1] **ENGE**:

0: Valid engine execution.

1: Invalid engine execution.

The following conditions set this bit:

– STCU2 state machines (watchdog, master and CBIST/MBIST controller) select an unused code.

– Serial MBIST global BEND status and the direct BEND flag are not aligned.

– Serial MBIST global BBAD status and the direct BBAD flag are not aligned.

– Serial MBIST single BBAD of not selected memories is not correct.

– Serial MBIST single march start-end status of not selected memories is not correct.

– Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned.

– Serial bend status of the selected MBIST and the global BEND flag are not aligned.

[0] **INVP**:
0: Valid pointer value.
1: Invalid pointer value. This bit is set if pointer value is out of range.
Invalid pointer is asserted for the following cases in case of MBIST run:
– If the number of tests scheduled > Total number of memories.
– BIST_TYPE is not valid value (0x0000, 0x0001, 0x0010, 0x1111).
– BIST_NMCUT field value > NMCUT (Total number of memories).
– BIST_CNTX_NUM field value > NMBIST (Total number of MBIST controllers).
In all other cases, INVP field is not set.

STCU_ERR_FM STCU error fault mapping register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											LOCKEUFM	WDTOUFM	CRCSUFM	ENGEUFM	INVPUFM
R																											RW	RW	RW	RW	RW

Address: STCU3BaseAddress + 0x0054

Type: RW

Reset: 0x00000000

Description: The STCU_ERR_FM register defines the fault mapping of the STCU3 faults described in the STCU_ERR_STATUS register in terms of unrecoverable or recoverable faults. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. However, these fields can be written into when either of the following conditions are true:

– Offline self-test phase is active.

[4] **LOCKEUFM**:
0: Recoverable fault mapping.
1: Unrecoverable mapping.

[3] **WDTOUFM**:
0: Recoverable fault mapping.
1: Unrecoverable mapping.

[2] **CRCSUFM**:
0: Recoverable fault mapping.
1: Unrecoverable mapping.

[1] **ENGEUFM**:
0: Recoverable fault mapping.
1: Unrecoverable mapping.

[0] **INVPUFM**:
0: Recoverable fault mapping.
1: Unrecoverable mapping.

55.5.2.2 MBIST configuration register description

STCU_MB0_TM STCU MBIST0 test mode selection register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														MBU	PMOSEN
R																														RW	RW

Address: STCU3BaseAddress + 0x0604

Type: RW

Reset: 0x00000000

Description: STCU test mode select register and can be updated both in offline testing.

[1] **MBU**:
0: MBIST full algorithm or reduced one without PMOS test (PMOSEN set to 0) is applied.
1: MBIST simplified multi-bit upset algorithm is used to check the RAM. If a ROM is selected, the applied algorithm is standard one without PMOS test. This bit overwrites the PMOSEN feature.

[0] **PMOSEN**:
0: MBIST PMOS test is not enabled.
1: MBIST PMOS test is enabled.

55.5.2.3 MBIST status register description

STCU_MB0_STATUS0 STCU offline MBIST0 status register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MB0_STATUS_31	MB0_STATUS_30	MB0_STATUS_29	MB0_STATUS_28	MB0_STATUS_27	MB0_STATUS_26	MB0_STATUS_25	MB0_STATUS_24	MB0_STATUS_23	MB0_STATUS_22	MB0_STATUS_21	MB0_STATUS_20	MB0_STATUS_19	MB0_STATUS_18	MB0_STATUS_17	MB0_STATUS_16	MB0_STATUS_15	MB0_STATUS_14	MB0_STATUS_13	MB0_STATUS_12	MB0_STATUS_11	MB0_STATUS_10	MB0_STATUS_9	MB0_STATUS_8	MB0_STATUS_7	MB0_STATUS_6	MB0_STATUS_5	MB0_STATUS_4	MB0_STATUS_3	MB0_STATUS_2	MB0_STATUS_1	MB0_STATUS_0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Address: STCU3BaseAddress + 0x0644

Type: R

Reset: 0x00000000

Description: The STCU_MB0_STATUS0 register includes the results corresponding to the execution of the selected offline MBIST0 in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST.

[31] **MB0_STATUS_31**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[30] **MB0_STATUS_30**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[29] **MB0_STATUS_29**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[28] **MB0_STATUS_28**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[27] **MB0_STATUS_27**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[26] **MB0_STATUS_26**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[25] **MB0_STATUS_25**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[24] **MB0_STATUS_24**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[23] **MB0_STATUS_23**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[22] **MB0_STATUS_22**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[21] **MB0_STATUS_21**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[20] **MB0_STATUS_20**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[19] **MB0_STATUS_19**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[18] **MB0_STATUS_18**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[17] **MB0_STATUS_17**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[16] **MB0_STATUS_16**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[15] **MB0_STATUS_15**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[14] **MB0_STATUS_14**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[13] **MB0_STATUS_13**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[12] **MB0_STATUS_12**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[11] **MB0_STATUS_11**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[10] **MB0_STATUS_10**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[9] **MB0_STATUS_9**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[8] **MB0_STATUS_8**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[7] **MB0_STATUS_7**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[6] **MB0_STATUS_6**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[5] **MB0_STATUS_5**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[4] **MB0_STATUS_4**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[3] **MB0_STATUS_3**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[2] **MB0_STATUS_2:**
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[1] **MB0_STATUS_1**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[0] **MB0_STATUS_0**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

STCU_MB0_STATUS1 STCU offline MBIST0 status1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						MB0_STATUS_57	MB0_STATUS_56	MB0_STATUS_55	MB0_STATUS_54	MB0_STATUS_53	MB0_STATUS_52	MB0_STATUS_51	MB0_STATUS_50	MB0_STATUS_49	MB0_STATUS_48	MB0_STATUS_47	MB0_STATUS_46	MB0_STATUS_45	MB0_STATUS_44	MB0_STATUS_43	MB0_STATUS_42	MB0_STATUS_41	MB0_STATUS_40	MB0_STATUS_39	MB0_STATUS_38	MB0_STATUS_37	MB0_STATUS_36	MB0_STATUS_35	MB0_STATUS_34	MB0_STATUS_33	MB0_STATUS_32
R						R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Address: STCU3BaseAddress + 0x0648

Type: R

Reset: 0x00000000

Description: The STCU_MB0_STATUS1 register includes the results corresponding to the execution of the selected offline MBIST0 in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST.

[25] **MB0_STATUS_57**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[24] **MB0_STATUS_56**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[23] **MB0_STATUS_55**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[22] **MB0_STATUS_54**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[21] **MB0_STATUS_53**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[20] **MB0_STATUS_52**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[19] **MB0_STATUS_51**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[18] **MB0_STATUS_50**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[17] **MB0_STATUS_49**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[16] **MB0_STATUS_48**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[15] **MB0_STATUS_47**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[14] **MB0_STATUS_46**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[13] **MB0_STATUS_45**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[12] **MB0_STATUS_44**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[11] **MB0_STATUS_43**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[10] **MB0_STATUS_42**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[9] **MB0_STATUS_41**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[8] **MB0_STATUS_40**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[7] **MB0_STATUS_39**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[6] **MB0_STATUS_38**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[5] **MB0_STATUS_37**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[4] **MB0_STATUS_36**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[3] **MB0_STATUS_35**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[2] **MB0_STATUS_34:**
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[1] **MB0_STATUS_33**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

[0] **MB0_STATUS_32**:
0: Failed NMCUT BIST execution.
1: No fault detected during the NMCUT BIST execution.

STCU_MB0_ENDFLAG0 STCU offline MBIST0 endflag register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MB0_ENDFLAG_31	MB0_ENDFLAG_30	MB0_ENDFLAG_29	MB0_ENDFLAG_28	MB0_ENDFLAG_27	MB0_ENDFLAG_26	MB0_ENDFLAG_25	MB0_ENDFLAG_24	MB0_ENDFLAG_23	MB0_ENDFLAG_22	MB0_ENDFLAG_21	MB0_ENDFLAG_20	MB0_ENDFLAG_19	MB0_ENDFLAG_18	MB0_ENDFLAG_17	MB0_ENDFLAG_16	MB0_ENDFLAG_15	MB0_ENDFLAG_14	MB0_ENDFLAG_13	MB0_ENDFLAG_12	MB0_ENDFLAG_11	MB0_ENDFLAG_10	MB0_ENDFLAG_9	MB0_ENDFLAG_8	MB0_ENDFLAG_7	MB0_ENDFLAG_6	MB0_ENDFLAG_5	MB0_ENDFLAG_4	MB0_ENDFLAG_3	MB0_ENDFLAG_2	MB0_ENDFLAG_1	MB0_ENDFLAG_0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Address: STCU3BaseAddress + 0x06C4

Type: R

Reset: 0x00000000

Description: The STCU_MB0_ENDFLAG0 register includes the end flag respective to the execution of the selected offline MBIST0 in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST.

[31] **MB0_ENDFLAG_31**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[30] **MB0_ENDFLAG_30**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[29] **MB0_ENDFLAG_29**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[28] **MB0_ENDFLAG_28**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[27] **MB0_ENDFLAG_27**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[26] **MB0_ENDFLAG_26**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[25] **MB0_ENDFLAG_25**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[24] **MB0_ENDFLAG_24**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[23] **MB0_ENDFLAG_23**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[22] **MB0_ENDFLAG_22**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[21] **MB0_ENDFLAG_21**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[20] **MB0_ENDFLAG_20**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[19] **MB0_ENDFLAG_19**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[18] **MB0_ENDFLAG_18**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[17] **MB0_ENDFLAG_17**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[16] **MB0_ENDFLAG_16**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[15] **MB0_ENDFLAG_15**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[14] **MB0_ENDFLAG_14**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[13] **MB0_ENDFLAG_13**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[12] **MB0_ENDFLAG_12**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[11] **MB0_ENDFLAG_11**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[10] **MB0_ENDFLAG_10**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[9] **MB0_ENDFLAG_9**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[8] **MB0_ENDFLAG_8**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[7] **MB0_ENDFLAG_7**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[6] **MB0_ENDFLAG_6**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[5] **MB0_ENDFLAG_5**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[4] **MB0_ENDFLAG_4**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[3] **MB0_ENDFLAG_3**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[2] **MB0_ENDFLAG_2**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[1] **MB0_ENDFLAG_1**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[0] **MB0_ENDFLAG_0**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

STCU_MB0_ENDFLAG1 STCU offline MBIST0 endflag1 register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						MB0_ENDFLAG_57	MB0_ENDFLAG_56	MB0_ENDFLAG_55	MB0_ENDFLAG_54	MB0_ENDFLAG_53	MB0_ENDFLAG_52	MB0_ENDFLAG_51	MB0_ENDFLAG_50	MB0_ENDFLAG_49	MB0_ENDFLAG_48	MB0_ENDFLAG_47	MB0_ENDFLAG_46	MB0_ENDFLAG_45	MB0_ENDFLAG_44	MB0_ENDFLAG_43	MB0_ENDFLAG_42	MB0_ENDFLAG_41	MB0_ENDFLAG_40	MB0_ENDFLAG_39	MB0_ENDFLAG_38	MB0_ENDFLAG_37	MB0_ENDFLAG_36	MB0_ENDFLAG_35	MB0_ENDFLAG_34	MB0_ENDFLAG_33	MB0_ENDFLAG_32
R						R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Address: STCU3BaseAddress + 0x06C8

Type: R

Reset: 0x00000000

Description: The STCU_MB0_ENDFLAG1 register includes the end flag respective to the execution of the selected offline MBIST0 in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST.

[25] **MB0_ENDFLAG_57**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[24] **MB0_ENDFLAG_56**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[23] **MB0_ENDFLAG_55**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[22] **MB0_ENDFLAG_54**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[21] **MB0_ENDFLAG_53**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[20] **MB0_ENDFLAG_52**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[19] **MB0_ENDFLAG_51**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[18] **MB0_ENDFLAG_50**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[17] **MB0_ENDFLAG_49**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[16] **MB0_ENDFLAG_48**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[15] **MB0_ENDFLAG_47**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[14] **MB0_ENDFLAG_46**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[13] **MB0_ENDFLAG_45**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[12] **MB0_ENDFLAG_44**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[11] **MB0_ENDFLAG_43**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[10] **MB0_ENDFLAG_42**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[9] **MB0_ENDFLAG_41**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[8] **MB0_ENDFLAG_40**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[7] **MB0_ENDFLAG_39**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[6] **MB0_ENDFLAG_38**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[5] **MB0_ENDFLAG_37**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[4] **MB0_ENDFLAG_36**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[3] **MB0_ENDFLAG_35**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[2] **MB0_ENDFLAG_34**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[1] **MB0_ENDFLAG_33**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

[0] **MB0_ENDFLAG_32**:
0: MBIST execution still ongoing.
1: MBIST execution finished.

STCU_MB0_UFM0 STCU MBIST0 Unrecoverable FM register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MB0_UFM_31	MB0_UFM_30	MB0_UFM_29	MB0_UFM_28	MB0_UFM_27	MB0_UFM_26	MB0_UFM_25	MB0_UFM_24	MB0_UFM_23	MB0_UFM_22	MB0_UFM_21	MB0_UFM_20	MB0_UFM_19	MB0_UFM_18	MB0_UFM_17	MB0_UFM_16	MB0_UFM_15	MB0_UFM_14	MB0_UFM_13	MB0_UFM_12	MB0_UFM_11	MB0_UFM_10	MB0_UFM_9	MB0_UFM_8	MB0_UFM_7	MB0_UFM_6	MB0_UFM_5	MB0_UFM_4	MB0_UFM_3	MB0_UFM_2	MB0_UFM_1	MB0_UFM_0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Address: STCU3BaseAddress + 0x0844

Type: RW

Reset: 0x00000000

Description: The STCU_MB0_UFM0 register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST. The R/W fields in this register are readable at any time.

[31] **MB0_UFM_31**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[30] **MB0_UFM_30**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[29] **MB0_UFM_29**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[28] **MB0_UFM_28**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[27] **MB0_UFM_27**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[26] **MB0_UFM_26**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[25] **MB0_UFM_25**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[24] **MB0_UFM_24**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[23] **MB0_UFM_23**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[22] **MB0_UFM_22**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[21] **MB0_UFM_21**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[20] **MB0_UFM_20**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[19] **MB0_UFM_19**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[18] **MB0_UFM_18**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[17] **MB0_UFM_17**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[16] **MB0_UFM_16**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[15] **MB0_UFM_15**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[14] **MB0_UFM_14**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[13] **MB0_UFM_13**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[12] **MB0_UFM_12**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[11] **MB0_UFM_11**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[10] **MB0_UFM_10**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[9] **MB0_UFM_9**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[8] **MB0_UFM_8**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[7] **MB0_UFM_7**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[6] **MB0_UFM_6**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[5] **MB0_UFM_5**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[4] **MB0_UFM_4**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[3] **MB0_UFM_3**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[2] **MB0_UFM_2**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[1] **MB0_UFM_1**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[0] **MB0_UFM_0**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

STCU_MB0_UFM1 STCU MBIST0 Unrecoverable FM register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						MB0_UFM_57	MB0_UFM_56	MB0_UFM_55	MB0_UFM_54	MB0_UFM_53	MB0_UFM_52	MB0_UFM_51	MB0_UFM_50	MB0_UFM_49	MB0_UFM_48	MB0_UFM_47	MB0_UFM_46	MB0_UFM_45	MB0_UFM_44	MB0_UFM_43	MB0_UFM_42	MB0_UFM_41	MB0_UFM_40	MB0_UFM_39	MB0_UFM_38	MB0_UFM_37	MB0_UFM_36	MB0_UFM_35	MB0_UFM_34	MB0_UFM_33	MB0_UFM_32
R						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Address: STCU3BaseAddress + 0x0848

Type: RW

Reset: 0x00000000

Description: The STCU_MB0_UFM1 register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 32 to 57. The size of the register depends on the amount of RAM or ROM subject to BIST. The R/W fields in this register are readable at any time.

[25] **MB0_UFM_57**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[24] **MB0_UFM_56**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[23] **MB0_UFM_55**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[22] **MB0_UFM_54**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[21] **MB0_UFM_53**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[20] **MB0_UFM_52**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[19] **MB0_UFM_51**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[18] **MB0_UFM_50**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[17] **MB0_UFM_49**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[16] **MB0_UFM_48**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[15] **MB0_UFM_47**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[14] **MB0_UFM_46**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[13] **MB0_UFM_45**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[12] **MB0_UFM_44**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[11] **MB0_UFM_43**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[10] **MB0_UFM_42**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[9] **MB0_UFM_41**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[8] **MB0_UFM_40**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[7] **MB0_UFM_39**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[6] **MB0_UFM_38**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[5] **MB0_UFM_37**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[4] **MB0_UFM_36**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[3] **MB0_UFM_35**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[2] **MB0_UFM_34**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[1] **MB0_UFM_33**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

[0] **MB0_UFM_32**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

55.5.2.4 CBIST status register description

STCU_CB_STATUS **Offline STCU CBIST status register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CB_STATUS15	CB_STATUS14	CB_STATUS13	CB_STATUS12	CB_STATUS11	CB_STATUS10	CB_STATUS9	CB_STATUS8	CB_STATUS7	CB_STATUS6	CB_STATUS5	CB_STATUS4	CB_STATUS3	CB_STATUS2	CB_STATUS1	CB_STATUS0
R																R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Address: STCU3BaseAddress + 0x02F80

Type: R

Reset: 0x00000000

Description: The STCU_CB_STATUS register includes the results corresponding to the execution of the selected offline CBIST. The size of the register depends on the number of CBISTs.

[15:0] **CB_STATUS[15:0]**:
0: Failed CBIST execution.
1: Successful CBIST execution.

STCU_CB_ENDFLAG Offline STCU CBIST endflag register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CB_ENDFLAG15	CB_ENDFLAG14	CB_ENDFLAG13	CB_ENDFLAG12	CB_ENDFLAG11	CB_ENDFLAG10	CB_ENDFLAG9	CB_ENDFLAG8	CB_ENDFLAG7	CB_ENDFLAG6	CB_ENDFLAG5	CB_ENDFLAG4	CB_ENDFLAG3	CB_ENDFLAG2	CB_ENDFLAG1	CB_ENDFLAG0
R																R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Address: STCU3BaseAddress + 0x2F84

Type: R

Reset: 0x00000000

Description: The STCU_CB_EDFLAG register includes the end flag respective to the execution of the selected offline CBIST. The size of the register depends on the number of CBISTs.

[15:0] **CB_ENDFLAG[15:0]**:
0: CBIST execution not yet completed.
1: CBIST execution finished.

STCU_CB_UFM STCU CBIST unrecoverable fault mapping register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CB_UFM15	CB_UFM14	CB_UFM13	CB_UFM12	CB_UFM11	CB_UFM10	CB_UFM9	CB_UFM8	CB_UFM7	CB_UFM6	CB_UFM5	CB_UFM4	CB_UFM3	CB_UFM2	CB_UFM1	CB_UFM0
R																RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Address: STCU3BaseAddress + 0x2F90

Type: RW

Reset: 0x00000000

Description: The STCU_CB_UFM register defines the fault mapping of each CBIST in terms of unrecoverable or recoverable fault. The size of the register depends on the number of CBISTs.

The R/W fields in this register are readable at any time.

[15:0] **CB_UFM[15:0]**:
0: Recoverable fault mapping.
1: Unrecoverable fault mapping.

55.5.2.5 MBIST pointer register description

STCU_MBISTn_PTR STCU MBISTn pointer register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				BIST_TYPE				RESERVED				RESERVED(1)				RESERVED					BIST_PTR_VAL										BIST_CFG
R				RW				R				R				R					RW										RW

1. The software guarantees the default value ‘0’.

Address: STCU3BaseAddress + 0x3040 + n*0x4 (n=0 to 57)

Type: RW

Reset: 0x00000000

Description: STCU_MBISTn_PTR defines the logical pointer to the MBIST/CBIST to be scheduled. The next MBIST/CBIST is scheduled concurrently to the current one if the BIST_CFG bit is set to 1, otherwise it is scheduled sequentially to the completion of the current BIST execution. “n” indicates the total number of memories present.

[27:24] **BIST_TYPE**: Indicates type of BIST for next BIST test execution. It provides a way to jump to other BIST pointer register files.
01: MBIST
10: CBIST
Only MBIST/CBIST can be reached from MBIST.
Note: END flag value is 0xF.

[10:1] **BIST_PTR_VAL**: Pointer to the memory per MBIST controller for BIST execution. The size of this field depends on the total number of memory cuts.

[0] **BIST_CFG**:
0: Sequential mode.
1: Concurrent mode.
Note: Concurrent tests are started immediately, whereas sequential stops waiting the previous tests to be completed.

55.5.2.6 CBIST pointer register description

STCU_CBISTn_PTR STCU CBISTn pointer register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				BIST_TYPE				RESERVED				BIST_CTRL_NUM				RESERVED															BIST_CFG
R				RW				R				RW				R															RW

Address: STCU3BaseAddress + 0x4080 + n*0x4 (n=0 to 10)

Type: RW

Reset: 0x00000000

Description: STCU_CBISTn_PTR defines the logical pointer to the CBIST to be scheduled. The next CBIST is scheduled concurrently to the current one if the BIST_CFG bit is set to 1, otherwise it is scheduled sequentially to the completion of the current BIST execution. “n” indicates the CBIST collector number.

[27:24] **BIST_TYPE**: Indicates type of BIST for next BIST test execution. It provides a way to jump to other BIST pointer register files.

01: MBIST
10: CBIST
Only CBIST can be mentioned for next execution from CBIST.
Note: END flag value is 0xF.

[19:16] **BIST_CTRL_NUM**: Defines the logical pointer to the current CBIST partition for BIST execution. The size of this field depends on the number of CBISTs partitions present on the device.

[0] **BIST_CFG**:

0: Sequential mode.
1: Concurrent mode.
Note: Concurrent tests are started immediately, whereas sequential stops waiting the previous tests to be completed.

56 Tamper detection module (TDM)

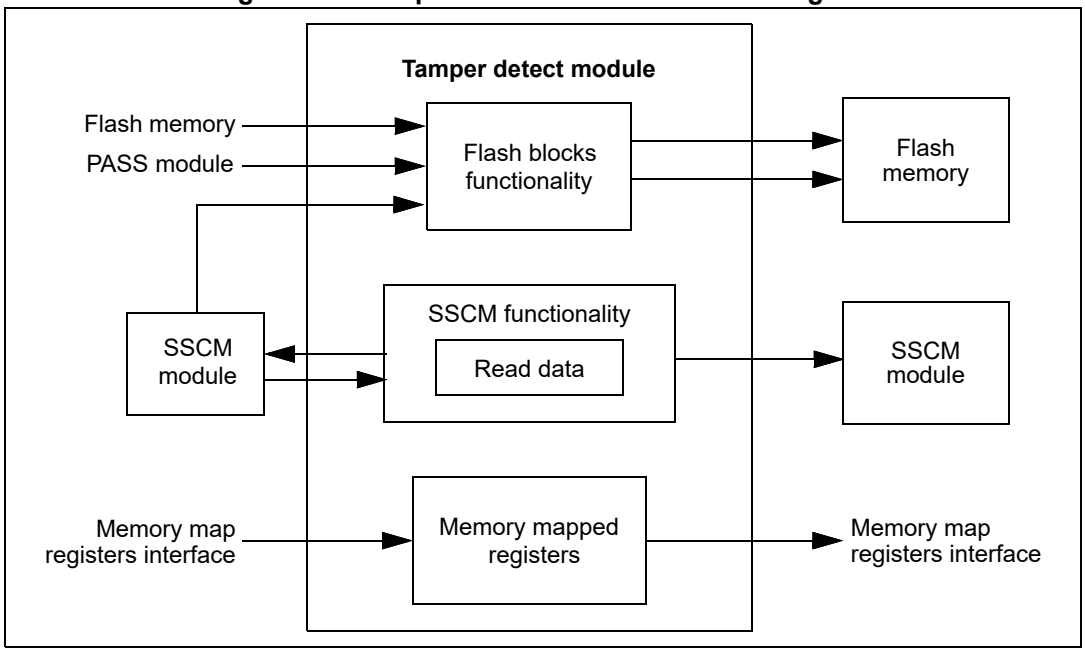
56.1 Overview

The tamper detection module provides a type of flash memory erase protection mechanism that forces software to write a record associated with one or more blocks in a tamper detection region (TDR) before the block(s) can be erased. The mechanism does not *prevent* an erase operation on any block within a TDR: it requires a record to be written before the operation can execute. Collectively, the records are referred to as a “diary”. At minimum, the diary serves as an erase counter. Because the diary record content is customer-defined, the diary can also serve as a history.

Up to 6 TDRs can be defined via DCF records.

Figure 958 shows the block diagram of the TDM and its interface to other system components.

Figure 958. Tamper detection module block diagram



Security information is passed to the TDM by the SSCM module. Among the data are:

- OTP (One-Time Programmable) settings for each of the flash blocks
- Base address for the diary in the flash
- Block assignments for the TDRs
- Tamper region override information (via the DCF bus)
- Software tamper region override

The TDM module controls the flash blocks for erase operations and sets them as OTP

The two flash buses are used to control the flash blocks for erase operation and set them as OTP. OTP means that flash erase of the entire block is disabled and the only words that are already erased can be programmed. Over-programming is not possible.

Note: The PASS module provides the control for erase operation for each flash block as well as for each region of the flash memory, and the PASS settings have priority over the settings of the TDM.

The flash block indicates to the TDM the address of the programmed word. This address is internally decoded and provides a means to verify whether the block is part of any TDRs. The memory map registers interface provides a means to the user to read information of the TDM registers.

56.2 Features

The Tamper detection module supports these distinctive features:

- Erase counter (diary) for each TDR to record up to 256 write attempts on the flash blocks
  - The diary is divided into 6 parts (TDRs), each consisting of 256 x 64-bit records for data flash (128 x 128-bit records for code flash)
  - The diary block must be assigned as OTP region in flash
  - The diary area can store any type of relevant information
- Tamper Detection Region Block Assignment
  - Any flash block can be assigned to any TDR and be controlled
- 32-bit Erase Lock DCF clients for each flash memory TDR defined
- 32-bit OTP Enable DCF clients to disable the erase operation for an entire flash block
- Override TDR via DCF records[g]
  - Allows block assigned to a TDR to be unlocked for erase even if the TDR is locked for this operation
- Memory Map Registers Interface for reading via software of specific registers

56.3 External signal description

There are no external signals driving or being driven by the TDM.

56.4 DCF client

TDM-related DCF records perform the following functions:

- Establish a permanent diary base address
- Define Tamper Detect Regions (TDRs) to monitor on-chip flash memory program/erase activity
- Permanently disable one or more Tamper Regions
- Permanently disable ability for software to permanently override one or more TDRs
- Configure flash memory as One Time Programmable (OTP) on a per-block basis

The following table gives an overview on the TDM DCF clients implemented.

g. The Tamper Region Override DCF record is a “write one only” DCF record. If an override is applied to a TDR it is permanent and cannot be reversed.

Table 691. DCF client address map

DCF Client Name	Description	Strategy
Diary base address	Stores the base address of the tamper detection diary	write-once, write 1 to 0 only
Tamper region override	Stores information of which TDR is to be overridden	write once, write 0 to 1 only
Software tamper region override Disable	Stores information to disable the software mechanism to override the TDR	write once, write 0 to 1 only
OTPEN[nb] (OTP enable)	Stores information of which flash block is to be assigned as OTP (One Time Programmable) - nb means one register for each Lock register of the flash memory	write 0 to 1 only
TDR[n]_LOCK[m] region assignment	Stores information of which flash block is assigned to a TDR – "*n*" specifies a TDR – "*m*" specifies the flash blocks as per the LOCK registers of the flash memory	write once, write 0 to 1 only

56.4.1 Diary base address (DBA) DCF client

This DCF client holds base address information of the diary. *Figure 959* shows the diagram for this register.

Caution: This DCF client is implemented as Write Once and is writable from 1 to 0 only. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0000 Access: DCF write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DBA															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DBA															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 959. Diary Base Address (DBA) DCF client

Table 692. DBA field descriptions

Field	Description
31:0 DBA	Diary Base Address This field holds the base address of the diary. This information is then used to verify if the diary being programmed matches any TDR region. **Note:** Diary Base Address must be on a 16KB boundary.

56.4.2 Tamper region override (TO) DCF client

This DCF client holds override information for each TDR. *Figure 960* shows the diagram for this register.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the override of a TDR permanent.

DCF Client Address: 0x0001 Access: DCF write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 960. Tamper Region Override (TO) DCF client

Table 693. TO field descriptions

Field	Description
5:0 TOE[5:0]	TDR Override Enable Each TOE field, where *n* represents the TDRs, holds information of the Override Enable status for each TDR. 0 Normal operation. Blocks assigned to this TDR can NOT be erased until the diary is written 1 Diary override. The diary for this TDR can be erased WITHOUT writing to the diary

56.4.3 One time programmable enable (OTPEN0) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK0 register of the flash memory. *Figure 961* depicts this client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash block to OTP irreversible.

DCF Client Address: 0x0008 Access: DCF write

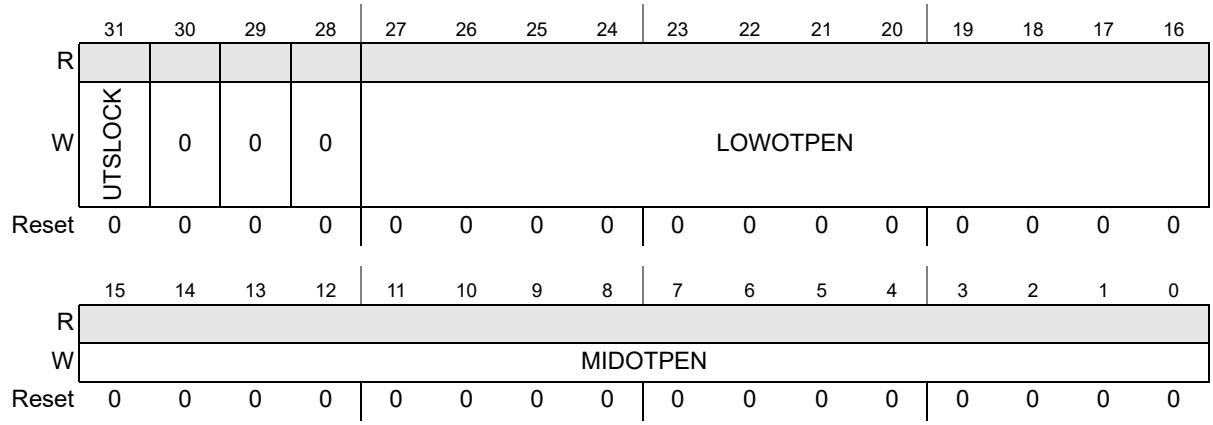


Figure 961. One Time Programmable Enable (OTPEN0) DCF client

Table 694. OTPEN0 field descriptions

Field	Description
31 UTSLOCK	UTEST NVM block OTP Enable This bit enables erase protection for the UTEST nonvolatile memory (NVM), making a flash memory block one-time programmable (OTP). It applies to flash memory blocks in the UTEST NVM address space: 0 The corresponding flash memory block is not OTP 1 The corresponding flash memory block is OTP
27:16 LOWOTPEN	Low Block OTP Enable This field affects flash memory blocks in Low address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP
15:0 MIDOTPEN	Mid Block OTP Enable This field affects flash memory blocks in Mid address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

56.4.4 One time programmable enable (OTPEN1) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK1 register of the flash memory. *Figure 962* depicts the client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash block to OTP irreversible.

DCF client Address: 0x0009 Access: DCF write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	HIGHOTPEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 962. One Time Programmable Enable (OTPEN1) DCF client

Table 695. OTPEN1 field descriptions

Field	Description
15:0 HIGHOTPEN	High Block OTP Enable This field affects flash memory blocks in High address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. (Refer to the embedded memories chapter). 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

56.4.5 One time programmable enable (OTPEN2) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK2 register of the flash memory. *Figure 963* depicts the client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash block to OTP irreversible.

DCF Client Address: 0x000A Access: DCF write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	256KBOTPEN[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	256KBOTPEN[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 963. One Time Programmable Enable (OTPEN2) DCF client

Table 696. OTPEN2 field descriptions

Field	Description
31–0 256KBOTPEN	256 KB Block OTP Enable (256KBOTPEN[31:0]) This field affects flash memory blocks in 256KB address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

56.4.6 One time programmable enable (OTPEN3) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF Client mirrors the LOCK3 register of the flash memory. *Figure 964* depicts the client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash block to OTP irreversible.

DCF client Address: 0x000B Access: DCF write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	256KBOTPEN[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 964. One Time Programmable Enable (OTPEN3) DCF client

Table 697. OTPEN3 field descriptions

Field	Description
15–0 256KBOTPEN	256 KB Block OTP Enable (256KBOTPEN[47:32]) This field affects flash memory blocks in the upper range of 256 KB address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

56.4.7 Tamper region assignment DCF client (TDR*n*_LOCK0)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF client however holds the assignment information for only those blocks specified in the LOCK0 register in the flash memory. Therefore, the diagram of this register mirrors the LOCK0 register in the flash memory. *Figure 965* depicts the TDR*n*_LOCK0 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash block to a TDR irreversible unless the entire TDR is overridden.

DCF client Address: 0x14 + ((n * 4) + 0) Access: DCF write

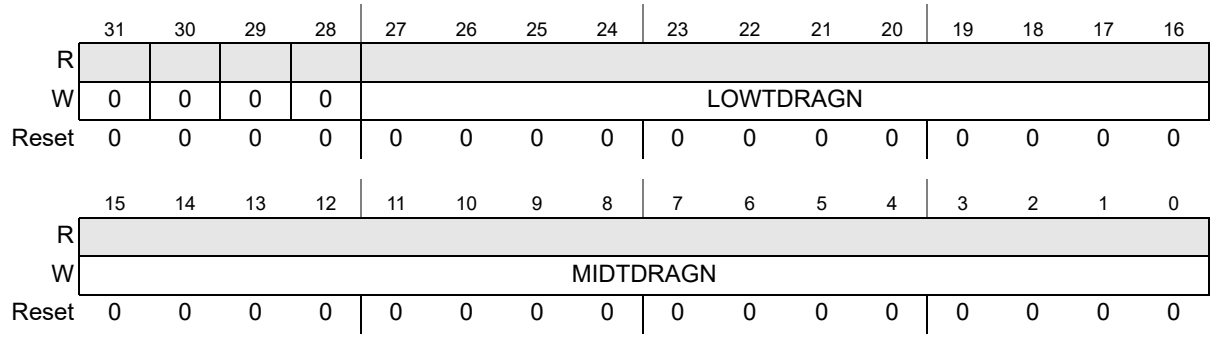


Figure 965. Tamper Region Assignment DCF client (TDR*n*_LOCK0)

Table 698. TDRn_LOCK0 field descriptions

Field	Description
27:16 LOWTDRAGN	Low Block Tamper Region Assignment This field affects flash memory blocks in Low address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*
15:0 MIDTDRAGN	Mid Block Tamper Region Assignment This field affects flash memory blocks in Mid address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

56.4.8 Tamper region assignment DCF client (TDR*n* _LOCK1)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF client however holds the assignment information for only those blocks specified in the LOCK1 register in the flash memory. Therefore, the diagram of this DCF client mirrors the LOCK1 register in the flash memory. *Figure 966* depicts the TDR*n*_LOCK1 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

DCF client Address: 0x14 + ((n * 4) + 1) Access: DCF write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	HIGHTDRAGN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 966. Tamper Region Assignment DCF client (TDR*n* _LOCK1)

Table 699. TDR*n* _LOCK1 field descriptions

Field	Description
15:0 HIGHTDRAGN	High Block Tamper Region Assignment This field affects flash memory blocks in High address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

56.4.9 Tamper region assignment DCF client (TDR*n*_LOCK2)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF Client however holds the assignment information for only those blocks specified in the LOCK2 register in the flash memory. Therefore, the diagram of this DCF Client mirrors the LOCK2 register in the flash memory. *Figure 967* depicts the TDR*n*_LOCK2 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash block to a TDR irreversible unless the entire TDR is overridden.

DCF client Address: 0x14 + ((n * 4) + 2) Access: DCF write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	256KBTDRAGN[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	256KBTDRAGN[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 967. Tamper Region Assignment DCF client (TDR*n*_LOCK2)

Table 700. TDR*n*_LOCK2 field descriptions

Field	Description
31:0 256KBTDRAGN	256 KB Block Tamper Region Assignment (256KBTDRAGN[31:0]) This field affects flash memory blocks in 256 KB address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

56.4.10 Tamper region assignment DCF client (TDR*n*_LOCK3)

This DCF Clients holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF Clients however holds the assignment information for only those blocks specified in the LOCK3 register in the flash memory. Therefore, the diagram of this DCF Clients mirrors the LOCK3 register in the flash memory. *Figure 968* depicts the TDRn_LOCK3 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash block to a TDR irreversible unless the entire TDR is overridden.

DCF client Address: 0x14 + ((n * 4) + 3) Access: DCF write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	256KBTDRAGN[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 968. Tamper Region Assignment DCF client (TDR*n*_LOCK3)

Table 701. TDR*n*_LOCK3 field descriptions

Field	Description
15:0 256KBTDRAGN	256 KB Block Tamper Region Assignment (256KBTDRAGN[47:32]) This field affects flash memory blocks in 256 KB address space. Refer to the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

56.5 Memory map and registers

Note: The Tamper Detection Module's memory-mapped registers are read-only. Any attempted write generates a transfer error.

Table 702. TDM memory map

Address offset (hex)	Register name	Location
0x00	TDR Status Register (TDM_TDRSR)	*Section 56.5.1*
0x04	Last Flash Programmed Address Register (TDM_LFPAR)	*Section 56.5.2*
0x08	Diary Base Address (TDM_DBA)	*Section 56.5.3*
0x10–0x24	Software Tamper Override Key Region (TDM_STO_KEY0–TDM_STO_KEY5)	*Section 56.5.4*

56.5.1 TDR status register (TDM_TDRSR)

This register contains the current status, such as locked or unlocked, of each tamper region, such as TDR.

- If a tamper region is locked, the blocks assigned to that tamper region cannot be erased.
- If a region is unlocked, the blocks within the tamper region can be erased.

The register contains a status bit for each TDR. Status is determined as follows:

1. The default status for each TDR is ‘1’, which indicates that the erase operation is disabled for all flash blocks associated with the TDR.
2. The status bit for a TDR is set to ‘0’ (indicating that the erase operation for all flash blocks associated with the TDR is enabled) if either a successful programming operation to the TDR diary region has been performed or the TDR Override bit for that TDR is set.

Note: TDRSR is a read-only register and writes have no effect. Write operations result in a transfer error.

If the reset value for this register is determined by a DCF record, it is noted in the chip-specific information at the beginning of this chapter. Otherwise, the reset value is as shown.

Offset: 0x00 Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	TDRSR5	TDRSR4	TDRSR3	TDRSR2	TDRSR1	TDRSR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 969. TDR Status and Diary Override Register (TDM_TDRSR)

Table 703. TDM_TDRSR field descriptions

Field	Description
5 TDRSR5	Represents the status of TDR 5 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
4 TDRSR4	Represents the status of TDR 4 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
3 TDRSR3	Represents the status of TDR 3 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
2 TDRSR2	Represents the status of TDR 2 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
1 TDRSR1	Represents the status of TDR 1 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
0 TDRSR0	Represents the status of TDR 0 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

56.5.2 Last flash programmed address register (TDM_LFPAR)

This register holds information of the address of the last successful programming operation provided from the flash memory to the TDM. Writes to this register are ignored and generate a transfer error.

Offset: 0x04 Access: User read

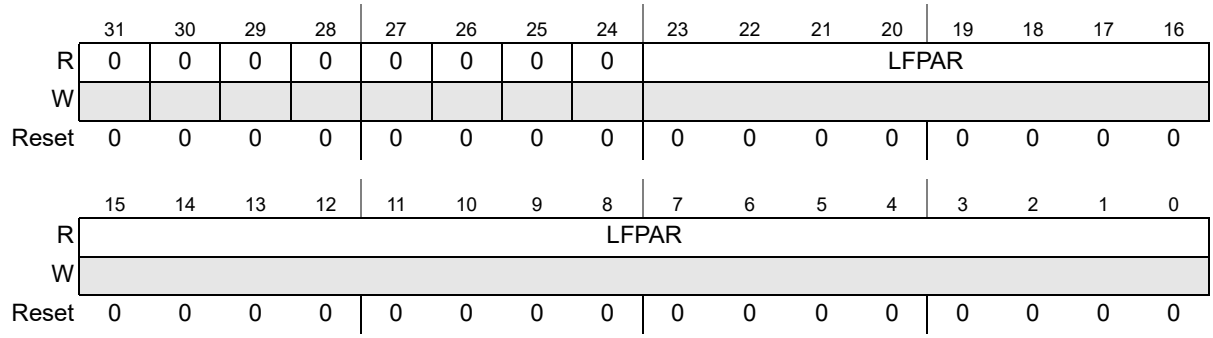


Figure 970. Last Flash Programmed Address Register (TDM_LFPAR)

Table 704. TDM_LFPAR field descriptions

Field	Description
23:0 LFPAR	Last Flash Programmed Address These bits represent the address of the last successful programming operation.

56.5.3 Diary base address (TDM_DBA)

This register holds base address information of the diary. Any write to this register is ignored and generates a transfer error.

Offset: 0x08 Access: User read

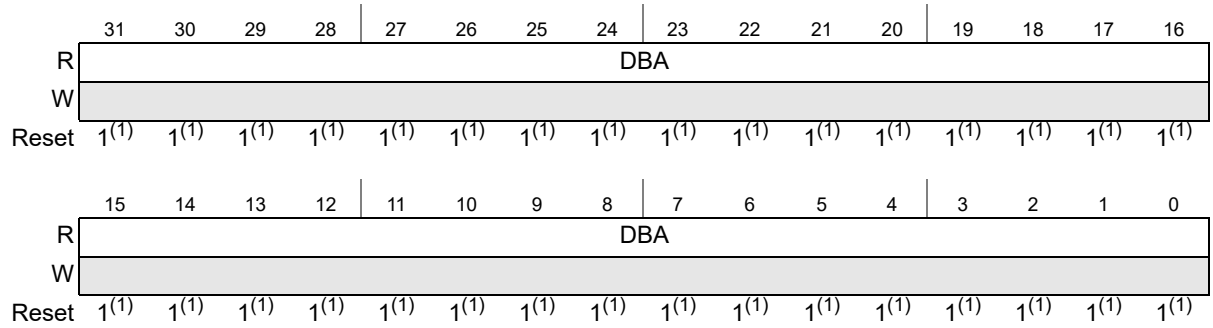


Figure 971. Diary Base Address Register (TDM_DBA)

1. Reset value depends on the Diary Base Address (DBA) DCF client.

Table 705. TDM_DBA field descriptions

Field	Description
31:0 DBA	Diary Base Address These bits represent the address of the diary location in the flash. **Note:** The diary base address is established via the DBA DCF record. The reset value of this register is determined by that record.

56.5.4 Software tamper override key region (TDM_STO_KEY*n*)

The STO_KEYn register holds the service key for overriding the Tamper Detect Region *n*, where *n* denotes the Tamper Detect Region number.

Caution: The STO_KEY*n* registers can only be written once.
The associated DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

Offset: 0x10 + *n* * 0x4 (*n* = 0 to 5) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	STO_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STO_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 972. Software Tamper Override Key Region (TDM_STO_KEY*n*)

Table 706. TDM_STO_KEY*n* field descriptions

Field	Description
31:0 STO_KEY	Software Tamper Override Key Diary *n* This field holds the service key value for overriding Tamper Detect Region n. By loading STO_KEYn with the correct signature of 0x55AA5A5A, Tamper Detect Region n is overridden, and the flash blocks assigned to Tamper Detect Region n are open for erase. This field has no effect when STO_DISn field of STO_DIS DCF client is set. **Note:** The STO_KEYn registers can only be written once.

56.6 Functional description

The primary purpose of the TDM is to provide a Tamper Detection mechanism by enabling an Erase Counter (or diary) for regions within the flash. These regions are called Tamper Detection Regions. The following sections cover the entire functionality of this module.

56.6.1 Flash erase counter and tamper detection

A flash region block or part of it is assigned for implementing the diary. The base address for the Dairy is defined by a DCF Record.

The diary is divided into 6 parts where each part consists of 256 x 64-bit records for data flash (128 x 128-bit records for code flash), to record until 256 write attempts on the flash blocks assigned to each Tamper Detection Region. For correct operation, the Diary must reside in a flash block that is assigned as OTP.

Each flash block can be assigned to any of the TDRs. Related to each TDR is a group of 4 × 32-bit internal hardware registers that are initialized by DCF records that define the tamper region. These registers mirror exactly the LOCK registers bits from the flash module.

Each TDRn_LOCK*m* register is loaded at boot time by the SSCM, which defines if the associated flash block is Locked for Erase in a particular TDR:

- ’1’ means the flash block is locked for erase
- ’0’ means the flash block is not locked for erase

Before performing an erase in one of the blocks locked for erase and associated to a TDR, a program operation has to be performed on the diary part for that TDR.As the diary has to be OTP, successive writes are allowed on unprogrammed double words only. This ensures that diary records can only increase and that data recorded in the diary is not modifiable by software.

At the end of a successful double word program operation, the flash module indicates to the TDM the address of the programmed double word. TDM determines whether the programmed data was written to the correct TDR Diary Location and then unlocks the blocks for erase operation for the corresponding TDR. TDM ensures that this erase operation on unlocked blocks is allowed until the next program operation on any block, other than Diary, is performed with program completion interrupt enabled or till next hard reset.

Note: All blocks associated to a TDR where an erase operation is allowed can be erased all together.

TDM has no control of how much data a diary has loaded. It is a responsibility for the user to verify if the diary is full. There is also no restriction to what type of data can be programmed in the diary. If diary is detected to be full, TDM provides a means to disable the tamper detect mechanism so that erase operations can be performed to a TDR without any diary entry. In this case, the user is required to program the Tamper Region Override Register for the corresponding TDR.

The data programmed in the diary location is flexible, and may include count information or other important customer information.

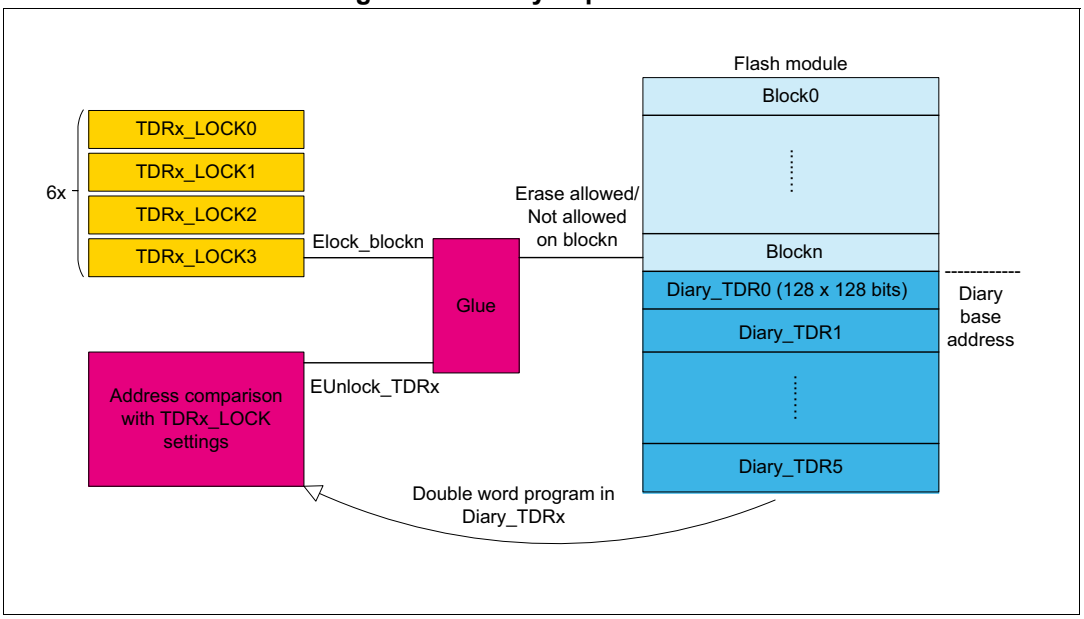
Software has to check whether the diary is full. Once the diary area for a TDR is full, the software has to program a DCF record for overriding blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry

The setting from “override tamper region lock” DCF record (refer to *Section 56.6.3: Diary*) has priority over the settings in the TDRn_LOCK0-3 registers. In other words, if a TDR is unlocked for erase by the override tamper region lock, the TDR*n*_LOCK*m* bits have no effect on the TDR that has been overridden, but the tamper detection remains active for all TDRs not having the override bit set.

Moreover, the settings from PASS_LOCK*x*_PG*n* registers have priority over TDRn_LOCK*m* settings. For example, if a block is password-protected for write operations, the TDR lock setting is ignored.

A schematic representation of the diary is depicted in *Figure 973*.

Figure 973. Diary implementation



Note: The settings in the Tamper Region Override Register have priority over the TDRn_LOCKm Registers. The SSCM module also has priority over the settings done in the TDM. For instance, if blocks in a TDR are unlocked for erase operation via TDM, but the same blocks are programmed to be locked for erase operation in SSCM, the blocks end up being locked for erase operation.

56.6.2 Assigning blocks to tamper detect regions (TDR)

Each TDR is defined by a group of 4 × 32-bit DCF records and clients, with each containing bits that map to specific blocks in a flash region. These mappings mirror exactly the LOCK registers bits from the flash module. Hence, the block assignment registers are named as TDRn_LOCKm where n is the number of a tamper region, that is, 0-5 and m varies from 0-3 as flash contains four Lock registers.

In these DCF records:

- ’1’ means that the corresponding flash block is assigned to this TDR
- ’0’ means that the corresponding flash block is not assigned to this TDR (Default)

DCF clients are not visible to software. They cannot be read nor written. They can only be initialized by DCF record within the UTEST flash block during reset. The value of these registers can be evaluated by reading all the UTest DCF records and searching for records that correspond to these clients.

Note: No flash block must be assigned to more than one TDR. Each block must correspond to a single TDR. For example, if a single block is assigned to TDR0 and TDR1, then even if the Diary is programmed for both TDR0 and TDR1, this flash block is never unlocked for erase operation.

56.6.3 Diary

The diary is a region of the flash memory where records of block erases for each TDR are stored. The format and size of the records are not fixed. The only requirements are that each diary has a maximum size of 2 KB, and the minimum size of any programming

operation depends on the Flash sector, which is 8 bytes for Data sector and 16 bytes for the remaining sector to honor OTP restrictions. Therefore, each diary can hold a maximum of 128 x 128-bit entries.

A flash region block or part of it must be assigned for implementing the diary. The base address for the diary is determined by a value written in a DCF record. The diary can be placed within any flash block in flash array, and history records in it can be read by any core.

Note: To maintain the security of the TDR, the block where the diary is placed must be assigned as OTP within the OTP registers.

The base address of the diary is implemented using a DCF Client. The DCF client is only writable once during reset and the default bit state is 1.

The DCF clients cannot be read nor written by software. They can only be initialized by DCF record from within UTEST block during reset. The value of these registers can be evaluated by reading all the UTEST DCF records and searching for records that write these DCF clients.

56.6.4 Specifying the diary base address

The diary is a region of flash memory where records of block erases for each TDR are stored. The format and size of these records are defined by the customer. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation depends on the Flash sector, which is 8 bytes for Data sector and 16 bytes for the remaining sector.

The diary can be placed within any flash block in the flash array. To maintain the security of the TDR, the block where the diary is placed, must be assigned as OTP within the OTP registers, while in customer delivery or OEM production life cycle states.

The base address of the diary is implemented with a 20-bit DCF client that can be initialized by the SSCM during reset. The DCF client is writable once during reset and the default bit state is ‘0’.

The base address register is not visible to software. It cannot be read nor written. It can only be initialized by DCF record within UTest flash during reset. The value of this register can be evaluated by reading all the UTest flash DCF Records and searching for record that writes this register.

Each TDR is assigned a 2 KB section of the diary. With 6 TDRs, 12 KB of diary space is required.

The diary base address must be at a 16 KB boundary. Therefore the least significant 14 bits of the base address must all be ‘0’.

The Base address of the diary is represented by:

Diary_Base_Address = 0bxxxx_xxxx_xxxx_xxxx_xxxx_0000_0000_0000

The base address of each TDR is shown below:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b10_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b10_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

56.6.5 Diary base address verification

The status of whether a TDR is locked for erase is maintained by the TDM. This is accomplished by implementing a register with as many bits as the number of TDRs. The default state of these bits is 1, which is defined as “TDR Erase Blocked”. Refer to *Section 56.5.1: TDR status register (TDM_TDRSR)* for further details about this register.

When the flash memory module has completed a programming operation, the address of the location programmed is output from the flash module. The TDM in turn compares that address to determine if it falls within the range of any TDR diary regions. If it is within any TDR, the corresponding status bit is set to 0 meaning “TDR Erase Enabled”.

These status bits are all reset to 1 when the flash module signals an erase complete by asserting the PEC (Program Erase Complete) signal.

The state of the 6 status bits can be read through the TDRSR. Refer to *Section 56.5.1: TDR status register (TDM_TDRSR)*. The address of the last successful programming operation provided from the flash memory to the TDM can be read through the LFPAR. Refer to *Section 56.5.2: Last flash programmed address register (TDM_LFPAR)*.

56.6.5.1 Diary management

Before a block protected by a tamper detect region can be erased, a record must be written to the associated diary. Software must search through the diary to find the end of the records, where the next available diary location can be written. After a maximum of 256 records for data flash (128 records for code flash) are programmed to the diary, a TDR diary is full. Full means that at least 1-bit has been programmed to ‘0’ in each record within the 2 KB diary.

If software determines that a particular TDR diary is full it may override the diary to allow erasing to continue or it may choose to stop further erases and to flag an error.

Diary Override bits are implemented with a DCF client consisting of 6 bits, one for each TDR. Refer to the device configuration information at the beginning of this chapter for TDM DCF client details.

The Diary Override bits are by default set to ‘0’; which means the diary must be written before an erase can occur. The DCF client can be written many times but the bits can only be written to ‘1’. Writes to ‘0’ are ignored. In this way, by default, all Tamper Detect Regions are active. The Tamper detect region diaries can be individually overridden by writing a DCF record that sets that corresponding bit to ‘1’. In this mode, the TDR status bit is permanently set to 0 to allow flash erase.

56.6.6 One time programmable (OTP)

56.6.6.1 Overview

Any flash block within the flash array can be assigned, at any time, to be OTP. Once a flash block is assigned OTP, it cannot be changed back.

OTP means that flash erase of the entire block is disabled and that only 64-bit double words that are already erased (that is, flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block becomes OTP after the next reset.

Appendix A Acronyms and abbreviations

Table 707 lists acronyms and abbreviations used in this document.

Table 707. Acronyms and abbreviations

Term	Meaning
AHB	Advanced High-performance Bus
AIC	Pbridge (Aips) Integrity Checker
AMBA	Advanced Microcontroller Bus Architecture
AMU	Asc Modeling Unit
AP	Access Port
APB	Advanced Peripheral Bus
ASILD	Automotive Sil D
ATB	Amba Trace Bus
AUTOSAR	Automotive Open System Architecture
AXI	Advance Extensible Interface
BAF	Boot Assist Flash
BIST	Built-in Self Test
BIU	Bus Interface Unit
BW	Bandwidth
CAN	Controller Area Network
CBIST	Comparator Built-In Self Test
CEM	Combined Error Management
CF	Critical Fault
CMU	Clock Monitoring Unit
CPU	Central Processing Unit
CQM	Clock Quality Monitor
CRC	Cyclic Redundancy Check
CTU	Cross Triggering Unit
DAP	Debug Access Port
DBE	Double Bit Error
DBGMCU	Debug Micro Controller Unit
DCF	Device Configuration Format
DMA	Direct Memory Access
DMAMUX	Direct Memory Access Controller Multiplexer
DMU	DMA Interface Unit
DP	Debug Port

Table 707. Acronyms and abbreviations (continued)

Term	Meaning
ECC	Error Correction Code
ECSM	Error Correction Status Module
EIM	Error Injection Module
EMI	Electromagnetic Interference
EOUT	Error Out
ETM	Embedded Trace Macrocell
ETR	Embedded Trace Router
FCCU	Fault Collection And Control Unit
(V)FEI	(Virtual) Fault Event Interrupt
FMPLL	Frequency-modulated Phase-locked Loop
FOSU	Fccu Output Supervision Unit
FPEC	NVM Program Erase Controller
FSM	Finite State Machine
FTTI	Fault Tolerant Time Interval
GIC	Generic Interrupt Controller
GPIO	General-purpose I/o
HVD	High Voltage Detector
HW	Hardware
IEEE	Institute Of Electrical And Electronics Engineers
Intf	Interface
IP	Intellectual Propriety
IPS	– Internal Peripheral System – Ipi Slave Bus
IRQ	Interrupt Request
ISR	Interrupt Service Routine
ITM	Instrumental Trace Macrocell
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LIN	Local Interconnect Network
LLDP	Link Layer Discovery Protocol
LVD	Low Voltage Detector
LWB	Late-write Buffer
MB	Message Buffer
MBIST	Memory Built-in Self Test
MBM	Message Buffer Management

Table 707. Acronyms and abbreviations (continued)

Term	Meaning
MBO	Message Box Overflow
MEMU	Memory Error Management Unit
MFD	Multiplication Factor Divider
Mux	Multiplex
NCF	Non-critical Fault
NMI	Non-maskable Interrupts
NVM	Non-volatile Memory
OPP	Over-program Protection
OSC	Oscillator
OTA	Over The Air
OTP	One Time Programmable
PASS	Password And Device Security Module
PLL	Phase-locked Loop
PMC	Power Management Controller
PPM	Passive Phase Modulation
PRAM	Platform RAM controller
PWD	Password
QoS	Quality of Service
RC	– Resistance-capacitance – Resistor-capacitor
RCCU	Redundancy Control Checker Unit
RF	Recoverable Fault
RFD	Reduced Frequency Divider
RMW	Read-modify-write
RTL	Register Transfer Language
RWW	Read-while-write
RX	Receive
SAG	Safety Application Guide
SAR ADC	Successive Approximation Register Analog-to-digital Converter
SBE	Single Bit Error
SIL	Safety Integrity Level
SoC	System-on-chip
SPI	Serial Peripheral Interface
STCU	Self-test Control Unit
SW	Software

Table 707. Acronyms and abbreviations (continued)

Term	Meaning
TBD	To be defined
TCD	Transfer Control Descriptor
TCM	Tightly Coupled Memory
TMC	Trace Macro Cell
TSENS	Temperature Sensor
TSU	Timestamping Unit
TX	Transmit
UART	Universal Asynchronous/synchronous Receiver Transmitter
UF	Unrecoverable Fault
VCO	Voltage Controlled Oscillator
VREG	Voltage Regulator
WDG	Watchdog timer
WS	Wait State

Revision history

Table 708. Document revision history

Date	Revision	Changes
11-Oct-2024	5	Chapter 2: Introduction Section 2.2: Features: in “AEC-Q100 automotive qualification” bullet, removed “on going” Chapter 5: Device configuration Section 5.5.9: RTC configuration: removed mention of “timestamp” Table 23: ADCx connectivity (x = 1..5): added note in channel type “Bias test” Table 29: HRTIM1 external events mapping and associated features and Table 30: HRTIM2 external events mapping and associated features: added the column “144-pin” and updated “176-pin” column Table 33: HRTIM1 DAC trigger connections and Table 34: HRTIM2 DAC trigger connections: updated hrtim_dac_step_trigx/ hrtim_dac_reset_trigx instances in alphabetic numbering (that is 1 -> A, 2 -> B...) Table 41: TIMX instance features overview: – prog. dead-time feature updated as “Yes: CH1” for TIM16 – timer input XOR function updated as “Yes” for TIM2-5 Table 72: Memory arrays accessible via IMA: updated “Function” column Chapter 6: Reset and boot Replaced occurrences of “C1_APPLI_ADDR_REG” with “C1_VTOR_ADDR_REG” Chapter 7: Device configuration format (DCF) records Section 7.3.1: UTEST DCF records: replaced “reset generation” with RCC Table 92: DCF client list: updated reset value of DCF client for UTEST miscellaneous Figure 19: DCF record structure: updated Chapter 10: Debug support (DBG) Table 106: Slave mapping: added

Table 708. Document revision history (continued)

Date	Revision	Changes
11-Oct-2024	5 (cont')	Chapter 11: Reset and clock control (RCC) Section 11.2: Clocks: updated UARTs clocks and RTC clock bullet points (replaced LSIRCOSC with LSICLK) Section 11.2.2: Internal RC oscillator (IRCOSC), Section 11.2.4: Low speed internal RC oscillator (LSIRCOSC) and Section 11.2.9: LSICLK clock branch: updated title of the sections Section 11.2.9: LSICLK clock branch: updated content Section 11.2.13: Internal/external clock measurement with TIM5/TIM15/TIM16: replaced "device datasheets" occurrences with a reference to the IO_definition document ICSR: replaced the register reset "long functional reset (PHASE1 reset)" with "power on reset (POR)" Table 108: Maximum IP clock frequencies: added Figure 46: Clock tree: added "LSICLK" Chapter 14: Power management controller digital interface (PMC_Dig) Table 153: GR_S field descriptions and Table 158: EPR_TD field descriptions: updated the temperature sensor point of bit fields TEMP_1 and TEMP_2 Chapter 17: Embedded flash memory Table 217: Flash memory EEPROM data partition 3 address mapping: fixed typo "0x009_C000" by "0x0090_C000" Table 192: UTEST block memory map: updated address offset and size of the UTEST DCF records Chapter 22: System memory protection unit (SMPU) Table 281: CESR0 field descriptions: added note in GVLD bit field Table 288: RGDn_WORD3 field descriptions: updated note in RO bit field Chapter 29: Analog-to-digital converters (ADC) Section 29.6.2.2: ADCx common control register (ADCx_CCR) (x = 1/2 or 3/4/5): put VREFEN bit field as reserved Chapter 31: Digital-to-analog converter (DAC) Section 31.1: Introduction and Section : Sample and hold mode: updated Section 31.4.1: DAC block diagram: updated note Section 31.4.13: DAC channel buffer calibration; removed content linked to VBAT Table 362: DAC2 interconnection: in "Signal name" column, updated all instances of "ch1" with "chx" and added "(x= 1 or 2)" Table 361: BDAC1 and DAC1 interconnection, Table 362: DAC2 interconnection, Table 363: DAC3 interconnection and Table 364: DAC4 interconnection: updated source and source type of the signal name dac_hold_ck and updated all instances of hrtimy_dac_step_trigx/ hrtimy_dac_reset_trigx in alphabetic numbering Table 366: HFSEL description: removed table note

Table 708. Document revision history (continued)

Date	Revision	Changes
11-Oct-2024	5 (cont')	Chapter 32: Comparator (COMP) Section 32.2.2: COMP pins and internal signals: replaced the reference to the datasheet with a reference to the IO-definition document Table 373: COMPx inverting input assignment — INMSEL: updated Table 375: Blanking sources: added row 0000 Chapter 34: High-resolution timer (HRTIM) Section 34.5.28: HRTIM timer x output 1 set register (HRTIM_SETx1R) (x = A to F): replaced “timer A” occurrences with “timer x” Section 34.5.29: HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F) and Section 34.5.31: HRTIM timer x output 2 reset register (HRTIM_RSTx2R) (x = A to F): added footnote Section 34.5.73: HRTIM fault input register 2 (HRTIM_FLTINR2): added value 11 in bit field FLT5SRC Table 379: HRTIM inputs/outputs summary: updated hrtim_dac_reset/step_trg in signal name column Table 395: HRTIM preloadable control registers and associated update sources: updated preloadable registers column Table 409: HRTIM register memory map: removed useless duplication Figure 336: High-resolution timer overview: replaced A:F occurrences by F:A Figure 338: Timer A..F overview and Figure 358: Master timer overview: updated Chapter 35: Advanced-control timers (TIM1/TIM8) Section 35.3.15: Combined PWM mode: updated channel 3 and channel 4 bullet points Figure 464: Combined PWM mode on channel 1 and 3: updated with the right channel numbers Chapter 36: General-purpose timers (TIM2/TIM3/TIM4/TIM5) Section 36.4.13: Combined PWM mode: updated channel 3 and channel 4 bullet points Section 36.4.21: Timer input XOR function: added content between parenthesis Figure 551: Combined PWM mode on channels 1 and 3: updated numbers Chapter 38: General-purpose timers (TIM15/TIM16) Section 38.4.13: Combined PWM mode (TIM15 only): updated configurations Figure 632: Combined PWM mode on channel 1: updated

Table 708. Document revision history (continued)

Date	Revision	Changes
11-Oct-2024	5 (cont')	Chapter 40: Real-time clock (RTC) Replaced LSE with LSI throughout the chapter Section 40.1: Introduction: removed content Section 40.2: RTC main features: – removed mention of VBAT pin – added reference to EXTI event input mapping table Section 40.3.2: RTC pins and internal signals, Section 40.3.5: Programmable alarm, Section 40.3.6: Periodic auto-wakeup and Section : RTC register write protection: removed content Section 40.3.13: Calibration clock output, Section 40.3.14: Tamper and alarm output and Section : TAMPALRM output: removed Section 40.6.8: RTC control register (RTC_CR): removed OUT2EN, TAMPALARM_TYPE, TAMPALARM_PU,TAMPOE, COE, OSEL, POL and COSEL bit fields Section 40.6.14: RTC status register (RTC_SR): removed ITSF, TSOVF and TSF bit fields Section 40.6.15: RTC masked interrupt status register (RTC_MISR): removed ITSMF, TSOVMF and TSMF bit fields Section 40.6.16: RTC status clear register (RTC_SCR): removed CITSF, CTSOVF and CTSF bit fields Table 458: RTC internal input/output signals: removed some signals Table 462: RTC interconnection and Table 464: RTC pins functionality over modes: removed Figure 661: RTC block diagram: updated Chapter 45: System window watchdog (WWDG) Section 45.3.4: How to program the watchdog timeout: removed reference to the datasheet Chapter 54: Indirect memory access (IMA) Section 54.2.3.4: IMA RAM Select register (IMA_SLCT): ROW_SLCT bit field updated as R/W Section 54.2.4.3.2: Unlocking reads and Section 54.2.4.3.3: Unlocking writes: reformulated the paragraph on how to make the READ/WRITE_KEY lock Figure 945: IMA block diagram: replaced 13-bit field by 15-bit field Chapter 55: Self-test control unit (STCU3) STCU_WDG: – updated reset value – in description, updated “15 to 0” by “31 to 0”
6-Apr-2026	6	*Chapter 1: Preface* *Table 1: Register conventions*: updated *Chapter 5: Device configuration* *Chapter 5.5.2.1: TIMx timer fault inputs* and *Chapter 5.8.4.2.1: FOSU parameters:* added

Table 708. Document revision history (continued)

Date	Revision	Changes
6-Apr-2026	6 (continued)	*Chapter 6: Reset and boot* Replaced occurrences of “Core1 User Application” with “Core1 User VTOR” *Chapter 7: Device configuration format (DCF) records* *Section 7.3: DCF records*: updated *Table 90: DCF record field description*, *Table 92: DCF client list*, *Table 93: UTEST Miscellaneous DCF Client field descriptions* and *Table 94: JTAG Pin Configuration field descriptions*: updated *Figure 20: DCF record structure*, *Figure 22: DCF stop record* and *Figure 25: JTAG Pin Configuration*: updated *Chapter 16: Nonvolatile memory platform controller (NVMPC)* *Section 16.1.2: Features*: updated *Section 16.9.2.4.2: OTA-X1 SWAP Error*: updated *Table 179: PFCR1 field descriptions*: added note in RWSC field description *Chapter 17: Embedded flash memory* *Section 17.4.5.1: Array Integrity Self Check*: updated *Table 219: UT0 field descriptions*: updated AIS field description *Chapter 18: Boot assist flash (BAF)* *Section 18.1: Introduction*, *Section 18.3.1: Boot modes*, *Section 18.3.5.1: ‘Soft’ DCF clients*, *Section 18.3.8: Optionally perform a serial boot*: updated *Figure 182: Flow of control*: updated *Chapter 21: System configuration controller (SYSCFG)* *HRTIMxFLT_TIMxBRK_EN*: updated TIMx_SAFEMODE_EN description *Chapter 26: Extended interrupt and event controller (EXTI)* *Section 26.6.10: Pending register (CnPR1)*: updated *Chapter 29: Analog-to-digital converters (ADC)* Removed content related to DEEPPWD throughout the chapter *Section 29.2: ADC main features* and *Section 29.4.6: Single-ended and differential input channels*: replaced V_{REFINT} with V_{REF} *Section 29.4.10: Channel selection (SQRx, JSQRx)*, *Section 29.6.1.3: ADC control register (ADC_CR)* and *DMA one shot mode (DMACFG = 0)*: updated *Chapter 30: Sigma-delta analog-to-digital converter (SDADC) digital interface* *Section 30.7.11: Watchdog*: updated

Table 708. Document revision history (continued)

Date	Revision	Changes
6-Apr-2026	6 (continued)	*Chapter 34: High-resolution timer (HRTIM)* *Section 34.5.16: HRTIM timer x DMA interrupt enable register (HRTIM_TIMxDIER) (x = A to F)*: updated *Figure 409: ADC trigger post-scaling in up/down counting mode*: updated *Chapter 35: Advanced-control timers (TIM1/TIM8)* *Figure 455: PWM input mode timing*: note added *Chapter 36: General-purpose timers (TIM2/TIM3/TIM4/TIM5)* *Figure 541: PWM input mode timing*: note updated *Chapter 38: General-purpose timers (TIM15/TIM16)* *Figure 626: PWM input mode timing*: note updated *Figure 627: Output compare mode, toggle on tim_oc1*, *Figure 628: Edge-aligned PWM waveforms (ARR=8)*: updated *Chapter 47: Controller area network (M_CAN)* *Section 47.2.3.3: Data bit timing and prescaler register (DBTP)*: updated *Section 47.2.3.7: Nominal bit timing and prescaler register (NBTP)*: updated description *Table 501: DBTP field descriptions*: updated *Chapter 53: Memory error management unit 2 (MEMU2)* *Section 53.4.5.1: Rationale*: updated *Table 645: SYS_RAM_CERR_STSn field descriptions*, *Table 656: PERIPH_RAM_CERR_STSn field descriptions*, *Table 667: NVM_SB_CERR_STSn register field descriptions*, *Table 670: NVM_DB_CERR_STSn register field descriptions*: updated *Chapter 56: Tamper detection module (TDM)* *Section 56.4.3: One time programmable enable (OTPEN0) DCF client*: updated

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers’ market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved