AN3260
Application note

Building a thermometer using the STM8S-DISCOVERY

Application overview

This application demonstrates how to build a simple thermometer based on the STM8S-DISCOVERY and the LM235 precision temperature sensor. The STM8S105C6T6 microcontroller reads the temperature values and transmits them through the UART interface. The temperature values are then displayed on a terminal window (possibly based on Windows HyperTerminal) of a PC connected to the UART through an RS232 cable.

Once the STM8S-DISCOVERY is powered up through an USB cable connected to the host PC, an informative message is displayed on the terminal window and the user is prompted to enter minimum and maximum temperature thresholds.

The current temperature is displayed on the terminal window every minute together with a warning message when the temperature is out of range.

The minimum and maximum values of the temperature over one-hour period are recorded in the MCU data EEPROM once per hour. They can be displayed any time by pressing a pushbutton.

Even if it is built on an STM8S105C6T6, the STM8S-DISCOVERY can be used to evaluate the main features of all STM8S microcontrollers.

Reference documents

- STM8S-DISCOVERY evaluation board user manual (UM0817).
- Developing and debugging your STM8S-DISCOVERY application code user manual (UM0834).
- LM235 precision temperature sensor datasheet
- ST232C 5 V powered multi-channel RS-232 driver and receiver datasheet

Contents

List of tables

List of figures

1 Prerequisites

The material required to run the STM8S-DISCOVERY thermometer demonstration application is the following:

- A terminal window running on a PC: the terminal emulator software can be Windows Hyperterminal (see *Appendix A*), TeraTerm Pro, or any terminal software.
- An RS232 null-modem cable (transmit and receive line crosslinked)
- USB type-A to mini-B cable.

2 Configuring the application

STM8S-DISCOVERY JP1 jumper must be set in 2/3 position to select a 5 V V_{DD} supply voltage.

3 Application description

3.1 Hardware required

This application uses the STM8S-DISCOVERY on-board LED (LD1) together with its associated resistor (R1).

The external passive components required by the application are listed in *Table 1*.

In addition, the application makes use of a 5 V powered ST232C RS232 driver/receiver. This extra component is essential since the COM port of the PC operates from a nominal 12 V power supply which is not compatible with the STM8S UART input/output operating at 5 V. This component is available in an SO16 package which fits the STM8S-Discovery footprint. Refer to the datasheet for more information on the ST232C. Refer to *Table 2* for the full list of packaged devices.

Table 1. List of passive components

Component name	Value	Comments
R2	2.2 kΩ	Pull-up resistor
R3, R4 (optional)	100 Ω	Current limitation resistors
C6, C7	100 nF	Debounce filters
Button1	-	Standard pushbutton
Button2	-	Standard pushbutton
C2, C3	100 nF	Charge-pump capacitors
C1, C4	100 nF	Output capacitors
C5	100 nF	Decoupling capacitor

Table 2. List of packaged components

Part number	Component description	Package
ST232C (order code ST232CN)	Very-high speed ultralow-power consumption 5 V RS232 driver/receiver used for UART 5/12 V level shifter.	SO16
LM235	Precision temperature sensor IC operating over a -40 to -125 °C temperature range with 1 °C initial accuracy.	SO8

3.2 Application schematics

Figure 1 shows how to interface the LM235 temperature sensor, the ST232C driver/receiver and the pushbuttons with the STM8S-DISCOVERY.

Button1 and button2 require an RC debounce filter to avoid triggering several interrupts. The debounce filter for button1 and button2 consist of C7 and C6 capacitors plus PA3 and PA4 internal pull-up resistor (about 45 KΩ).

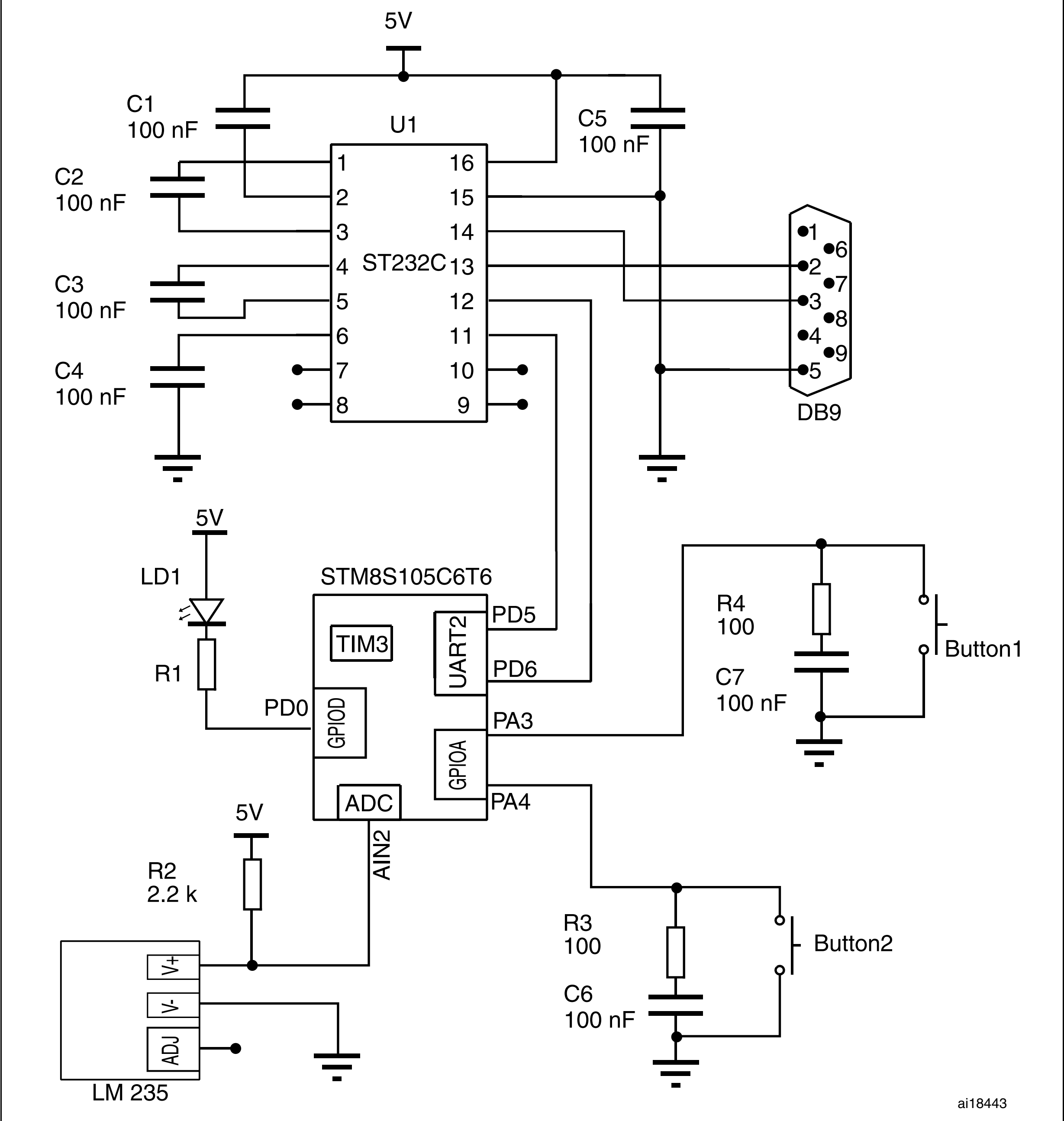
No external pull-up resistors are required as the internal pull-ups of the I/Os are used.

C2 and C3 are two charge-pump external capacitors which are used in the ST232C as voltage doubler and voltage inverter, respectively.

The current flowing into the LM235 V+ pin must be regulated by a resistor. The sensor is powered from a 5 V power supply (V_{DD}). The breakdown voltage across the sensor is directly proportional to the absolute temperature with a sensitivity of 10mV/°K. Since the ambient temperature is around 300 °K, the voltage drop is roughly 3 V, which leaves 2 V for the 2.2 kΩ resistor to regulate the current around 1 mA (intensity used to determine the typical values in the datasheet).

For implementation details, refer to the STM8S-DISCOVERY board schematics provided in the STM8S-DISCOVERY user manual (UM0817).

Figure 1. Application schematics



3.3 Application principle

At application startup, informative messages are displayed on the terminal window and the user is prompted to enter critical minimum and maximum temperature thresholds.

The LM235 continuously measures the ambient temperature. The analog value is converted by the STM8S105C6T6 ADC1 every 50 ms at each timer interrupt. To improve temperature measurement reliability, temperature data are obtained by averaging the first 16 samples measured after 1 second has elapsed. The resulting data are then compared to the current minimum and maximum temperature thresholds which can be modified if needed. LD1 is switched on if the temperature is below the low threshold or exceeds the high threshold defined by the user.

Once per minute, the last computed average temperature is displayed on the terminal window together with the critical temperature warning message when relevant.

The minimum and maximum temperatures over one hour period are recorded in the data EEPROM once per hour and displayed on the terminal window.

Pressing Button1 prompts the user to enter new temperature threshold values.

Pressing Button2 triggers the display on the terminal window of all recorded minimum and maximum temperatures stored in data EEPROM. LD1 is switched off.

Figure 2 shows the application state diagram and *Table 2* describes the actions performed by the application at each transition. The algorithm that controls the progress of the execution according to the timer and external events is managed by the State_Machine() function (see *Section 4.3.3: State machine flowchart*).

Figure 2. Application state machine

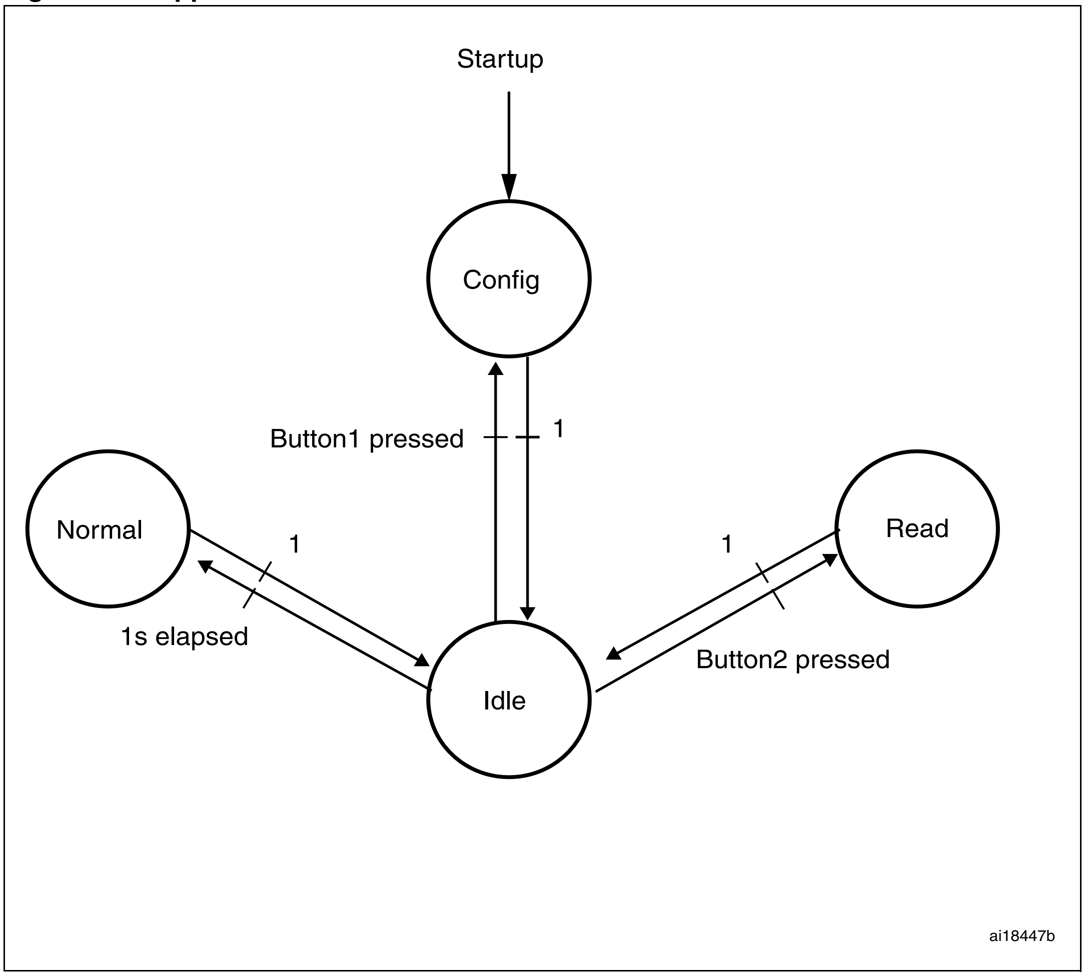


Table 3. Application typical behaviors

Application state	LED state	Entry condition	Actions
State 0 Idle	-	Default state	Every 50 ms: conversion of analog temperature delivered by the LM235.
State 1 Config	-	At startup or when Button 1 pressed	Informative messages displayed on the terminal window. User prompted to enter for the first time or to update minimum and maximum temperature threshold.
State 2 Read	LD1 switched off	Button 2 pressed	Minimum and maximum temperature values read from data EEPROM and displayed on terminal window
State 3 Normal	- LD1 switched on if temperature out of range - -	Every 1 s	– Every 1 s: Computation of average temperature from 16 samples. Update of current minimum and maximum temperature values if needed. – Every 1 min: display of current temperature value and critical temperature warning message on the terminal window (if need be). – Every 1 hr: minimum and maximum temperature over the previous hour recorded in data EEPROM.

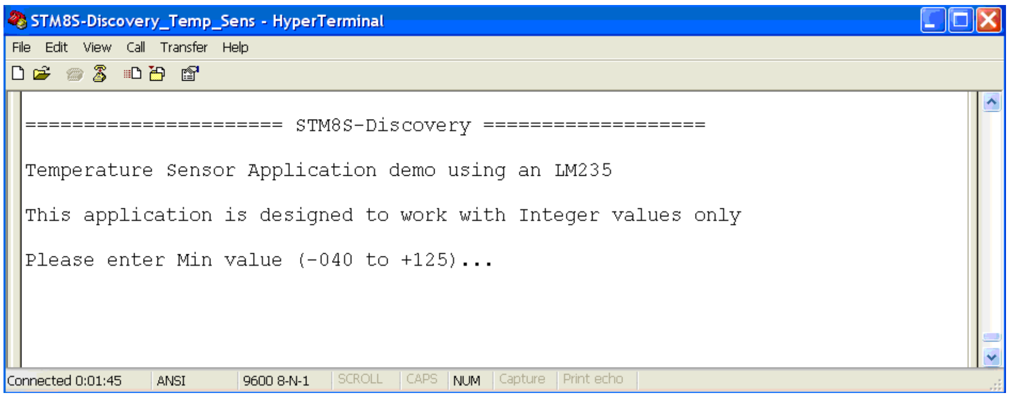
3.4 Launching your application from the PC terminal

To display the terminal window, you can either run the preconfigured STM8S-Discovery_Thermometer.ht terminal based on Windows HyperTerminal (COM1 port) or create one by proceeding as explained in *Appendix A*.

At application startup, the user is prompted to enter the minimum and maximum temperature thresholds. The temperature thresholds must range from –40 to +125 °C, and be expressed as follows:

- Positive temperature values: '+XXX'. For example enter '+025' for 25 °C.
- Negative temperature values: '–XXX'. For example enter '–005' for –5 °C.

Figure 3. Terminal window at startup


STM8S-Discovery_Temp_Sens - HyperTerminal
File Edit View Call Transfer Help

```
===================== STM8S-Discovery ===================

Temperature Sensor Application demo using an LM235

This application is designed to work with Integer values only

Please enter Min value (-040 to +125)...
```

Connected 0:01:45 ANSI 9600 8-N-1 SCROLL CAPS NUM Capture Print echo

4 Software description

4.1 STM8S peripherals used by the application

The thermometer application software does not use STM8S standard firmware library. It rather consists in optimized code by using direct register accesses to control and use the STM8S general purpose peripherals as described below:

- CLK

  The clock controller enables and delivers the correct clock frequency to the CPU and peripherals. It configures the HSE clock as the 16 MHz master clock source and the CPU clock prescaler division factor to 1.
- GPIOs

  The STM8S GPIOs are used to switch on and off LD1, and to configure PA3 and PA4 to interface with pushbuttons.
- EXTI

  The external interrupt sensitivity is configured to trigger an interrupt each time a falling edge and only a falling edge, is detected on PA3 or PA4.
- Flash memory

  The minimum and maximum temperature values over one-hour period are saved in the data EEPROM for further display on the terminal window.
- UART2

  UART2 is used to communicate with the PC terminal software. It is configured as follows:
  - Baud rate = 9600 baud
  - Word length = 8 bits
  - One stop bit
  - No parity
  - Receive and transmit enabled
  - UART2 clock disabled

  Communications are managed by polling each receive and transmit operation on the UART2 peripheral.
- ADC1

  Channel 2 of the ADC1 is used to convert the analog data issued from the LM235 to digital values out of which the microcontroller can compute the current temperature value.
- Timer 3 (TIM3)

  This peripheral is used to generate a 50-ms timebase and to trigger a temperature acquisition every 50 ms.

4.2 Exclusion of the Standard STM8S standard firmware library

As this application uses optimized code, the *stm8s.h* file must be modified not include the STM8S standard firmware library. This is done by commenting the following define statement:

```
#define USE_STDPERIPH_DRIVER
```

4.3 Application software flowcharts

4.3.1 Main loop flowchart

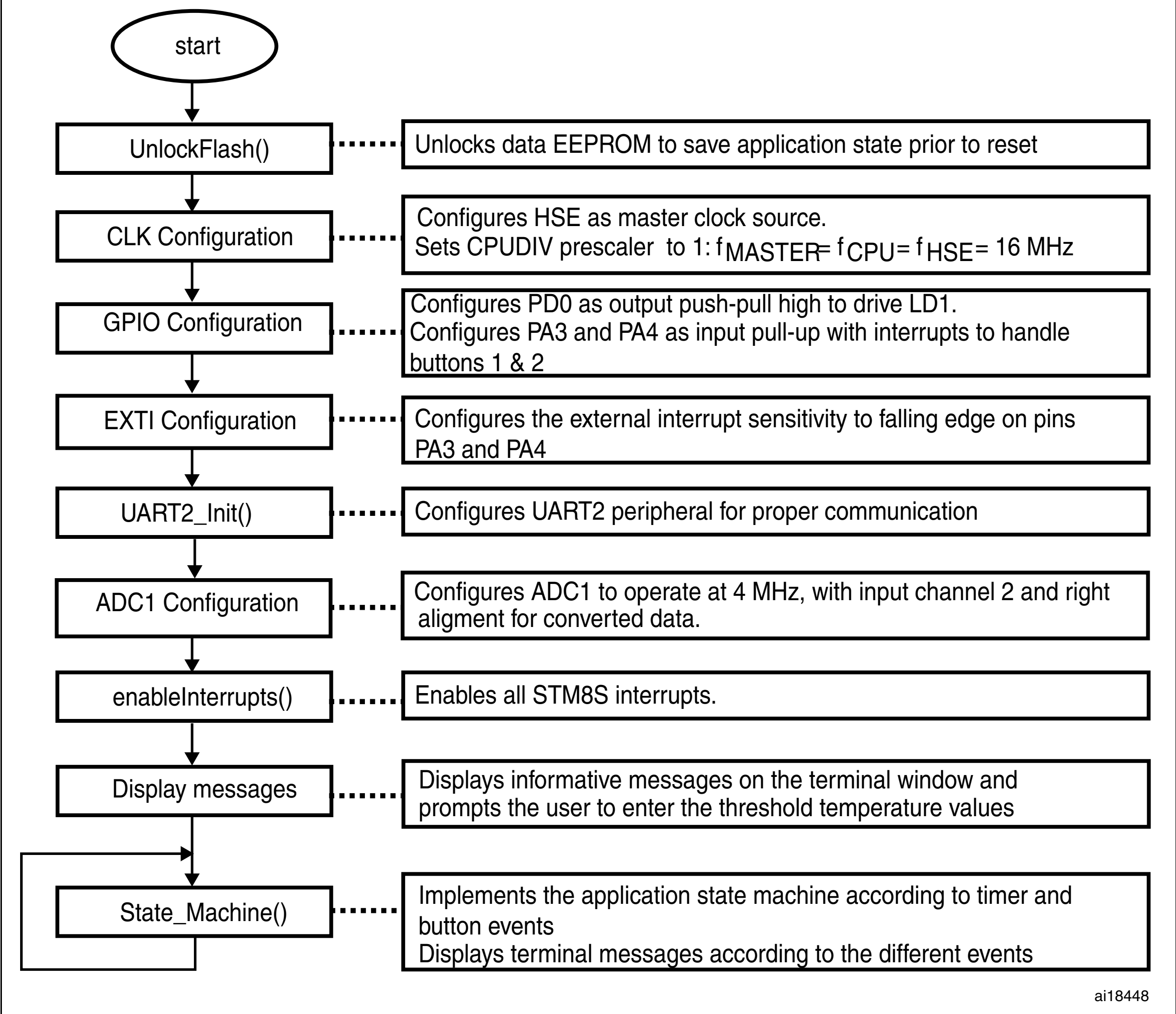
The main loop code initializes the required features, unlocks data EEPROM programming and calls the functions required to implement the general application algorithm.

To minimize the time drift of the software real time clock (see *Section : Timer-triggered acquisition*), the HSE clock is used as the master clock source.

After the initialization phase is complete and informative messages are displayed on the terminal window, the user is prompted to enter the minimum and maximum temperature thresholds. As a consequence, the first time the State_Machine() function is called, it directly enters the configuration mode (state = 1, see *Section 4.3.3: State machine flowchart*).

Figure 4 shows the flowchart of the application software main loop.

Figure 4. Main loop flowchart


start
UnlockFlash()
Unlocks data EEPROM to save application state prior to reset
CLK Configuration
Configures HSE as master clock source.
Sets CPUDIV prescaler to 1: fMASTER= fCPU= fHSE= 16 MHz
GPIO Configuration
Configures PD0 as output push-pull high to drive LD1.
Configures PA3 and PA4 as input pull-up with interrupts to handle buttons 1 & 2
EXTI Configuration
Configures the external interrupt sensitivity to falling edge on pins PA3 and PA4
UART2_Init()
Configures UART2 peripheral for proper communication
ADC1 Configuration
Configures ADC1 to operate at 4 MHz, with input channel 2 and right aligment for converted data.
enableInterrupts()
Enables all STM8S interrupts.
Display messages
Displays informative messages on the terminal window and prompts the user to enter the threshold temperature values
State_Machine()
Implements the application state machine according to timer and button events
Displays terminal messages according to the different events
ai18448

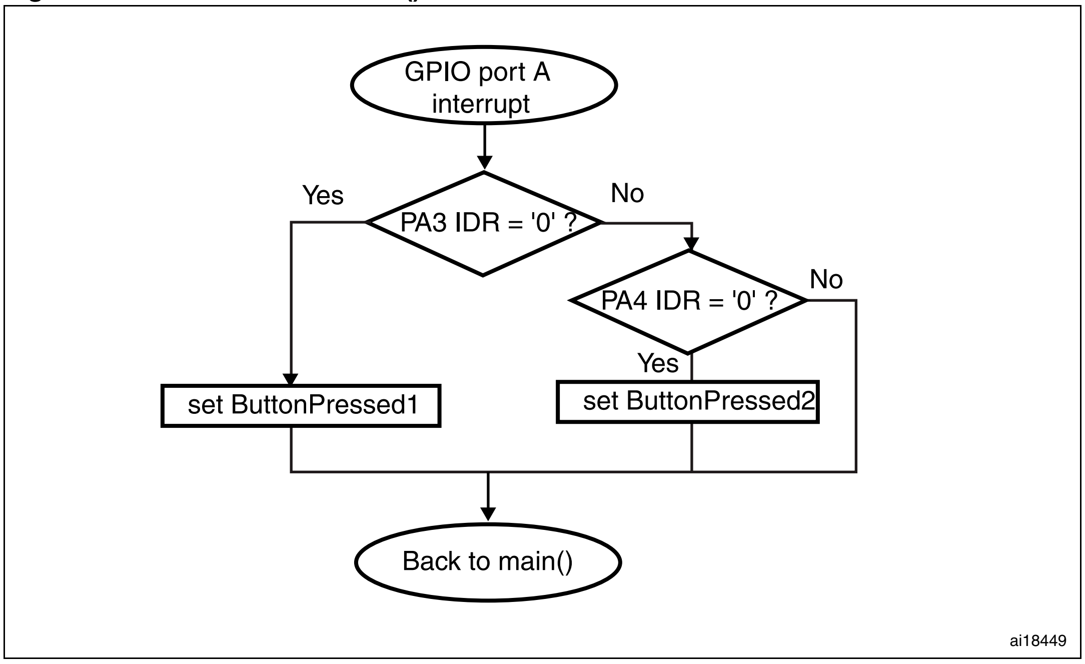
4.3.2 Interrupt function flowcharts

Pushbutton acquisition

Each time Button1 or Button2 is pressed, an interrupt is triggered and the PORTA_IRQhandler() function is called. The PORTA_IRQhandler() routine identifies which pushbutton has been pressed by testing port A input register and asserts the ButtonPressed1 or ButtonPressed2 flag accordingly (see *Section 4.3.3: State machine flowchart*). These flags trigger a change of state in the application state machine

Figure 5 shows the flowchart of the PORTA_IRQhandler() function.

Figure 5. PORTA_IRQhandler() function flowchart



Timer-triggered acquisition

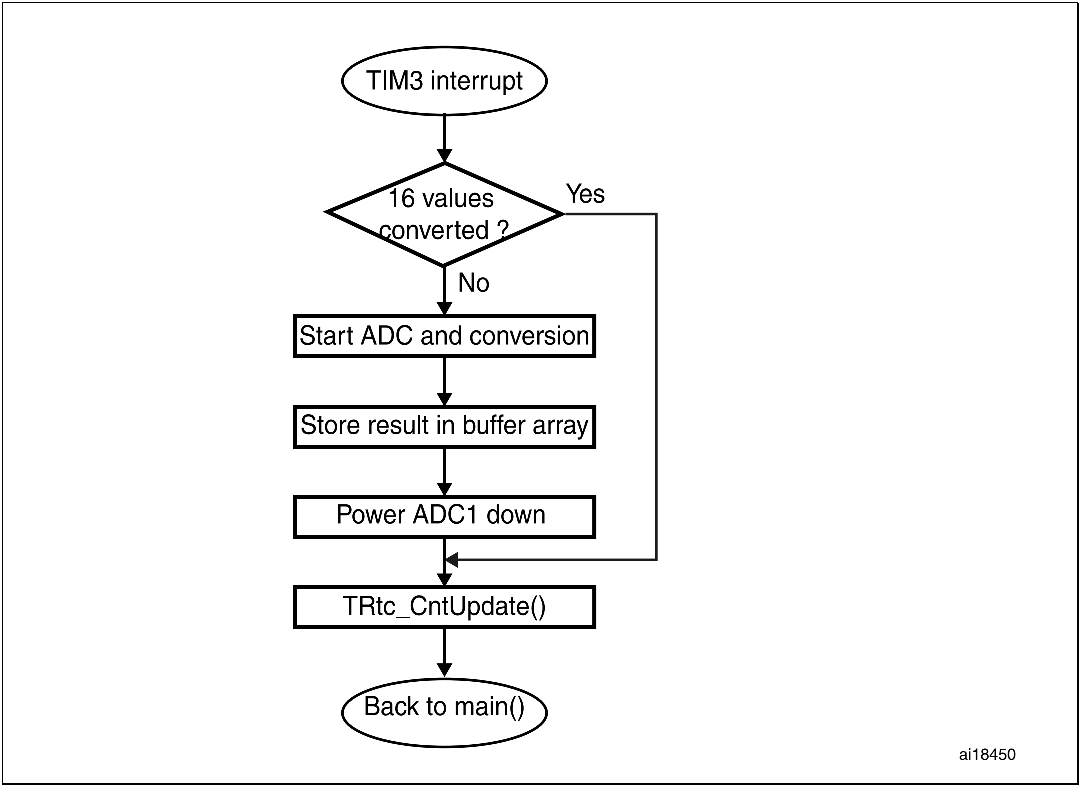
TIMER3 is configured to generate an interrupt every 50 ms to trigger temperature acquisitions.

After each conversion of ADC1 channel 2, the digital value is stored in the buffer array for further computations and ADC1 is powered down. A maximum of 16 samples are saved in the buffer array. This allows dividing by a power of two by performing a simple right shift, when calculating the average temperature over one-second period.

TIM3_IRQHandler() calls the TRtc_CntUpdate() function which simulate a real time clock. It sets the flags representing seconds, minutes, and hours. This RTC routine is based on TIM3 50 ms timebase. Every second, the state machine automatically switches to state 3 (normal mode) (see *Section 4.3.3: State machine flowchart*).

Figure 6 shows the flowchart of the TIM3_IRQhandler() function.

Figure 6. TIM3_Init() function flowchart



4.3.3 State machine flowchart

The State_Machine function implements the algorithm that controls the progress of the application execution according to timer and external events. The different values of the *state* variable represent the application modes.:

- state = 0: Idle mode

  This is the state machine default state. The conversion of the analog temperature delivered by the LM235 is performed every 50 ms. This state is exited by pressing button1 or button2.
- state = 1: Config mode

  In state machine configuration mode, a terminal message prompts the user to input the minimum and maximum temperature thresholds. These values are read from the terminal window and recorded in decimal format. This state is entered at startup or when button1 is pressed.
- state = 2: Read mode

  In state machine read mode, the application reads back the minimum and maximum temperature pairs from data EEPROM and displays them on the terminal window together with an informative message. LD1 is also switched off. This state can be entered only by pressing button2.
- state = 3: Normal mode

  The state machine normal mode is entered every second. It is triggered by the TRtc_CntUpdate() routine that monitors the application time elapsed. Each time this state is entered, the average temperature is computed from 16 samples saved in the buffer array. If this temperature value exceeds the high threshold or is below the low threshold, LD1 is switched on.

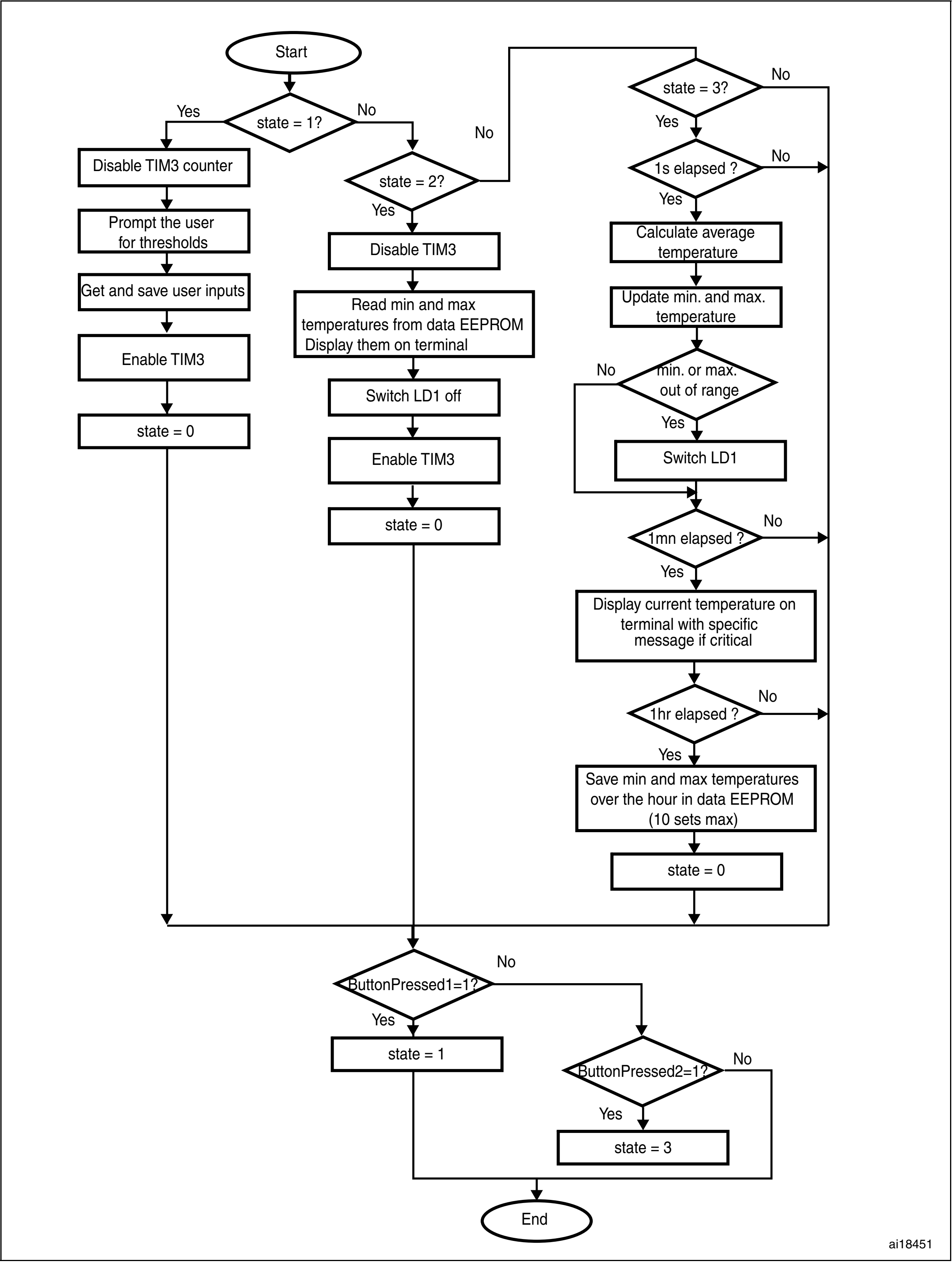
The current temperature is displayed on the terminal window once per minute together with a specific informative message when it is critical.

A pair of maximum and minimum measured temperatures is recorded in data EEPROM over one-hour period. Up to 10 pairs can be stored in EEPROM. This means that a history of ten hour measurements is kept. Writing the 11th hour data resets the EEPROM address pointer and overwrites the previous data. As a consequence, the data from the previous ten hours will be overwritten.

When entering states 1 and 2, the TIM3 counter is disabled as the normal mode is exited. When exiting these states, the timer registers are reassigned and the counter is enabled again to resume the default execution mode.

Figure 7 shows the flowchart of the State_Machine() function.

Figure 7. State_Machine function flowchart


Start
state = 1?
Yes
No
Disable TIM3 counter
Prompt the user for thresholds
Get and save user inputs
Enable TIM3
state = 0
state = 2?
No
Yes
Disable TIM3
Read min and max temperatures from data EEPROM Display them on terminal
Switch LD1 off
Enable TIM3
state = 0
state = 3?
No
Yes
1s elapsed ?
No
Yes
Calculate average temperature
Update min. and max. temperature
min. or max. out of range
No
Yes
Switch LD1
1mn elapsed ?
No
Yes
Display current temperature on terminal with specific message if critical
1hr elapsed ?
No
Yes
Save min and max temperatures over the hour in data EEPROM (10 sets max)
state = 0
ButtonPressed1=1?
No
Yes
state = 1
ButtonPressed2=1?
No
Yes
state = 3
End
ai18451

4.3.4 Terminal communication functions

For a detailed description of the terminal communication functions, refer to *Appendix A*.

Appendix A Configuring your terminal window

The terminal window connected to the STM8S-DISCOVERY must be configured with the following settings valid for all terminal types:

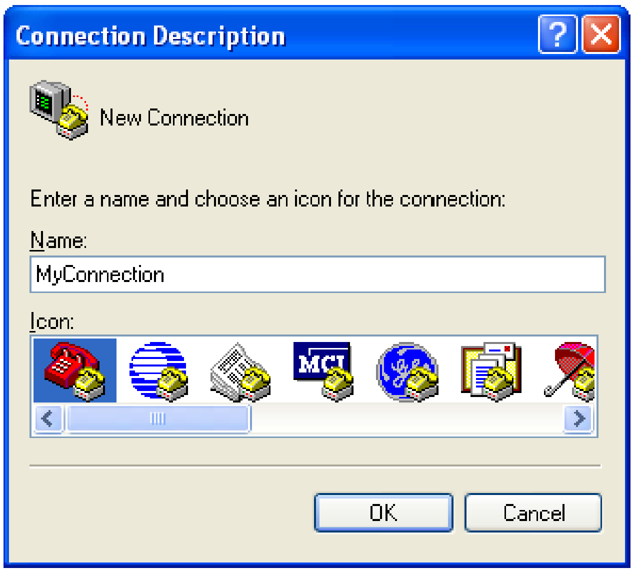
- Communication port: COM1 or other available
- Bits per second: 9600
- Data bits: 8
- Parity: none
- Stop bits: 1
- Flow control: none

To provide a ready-to-use application example, a preconfigured terminal using Windows HyperTerminal and COM1 port is provided within the project folder. To launch it, simply execute the .ht file included in the project.

However, you can also set up a new connection with the STM8S-DISCOVERY based on Windows HyperTerminal and related to this example by following the steps below:

1. Open Windows HyperTerminal application and choose a connection name, such as "MyConnection" and validate it by clicking **OK**.

Figure 8. Launching Windows HyperTerminal

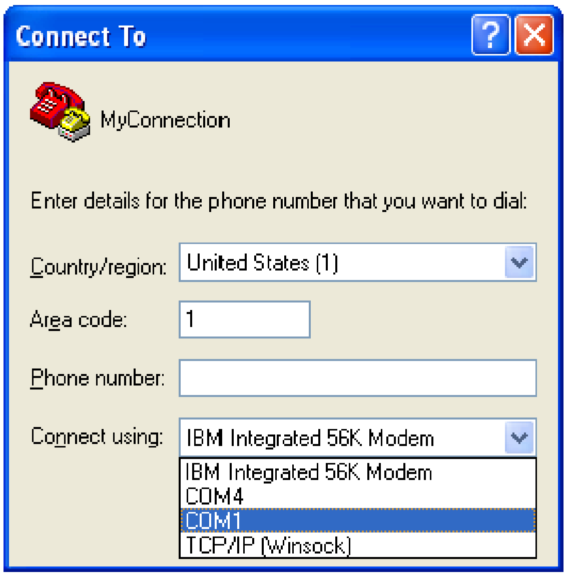


Enter a name and choose an icon for the connection:

Name: MyConnection

2. Select COM1 or any available port on your computer and validate your choice by clicking **OK**. Other fields can remain set to the default value.

Figure 9. Selecting communication port



Connect To

MyConnection

Enter details for the phone number that you want to dial:

Country/region: United States (1)

Area code: 1

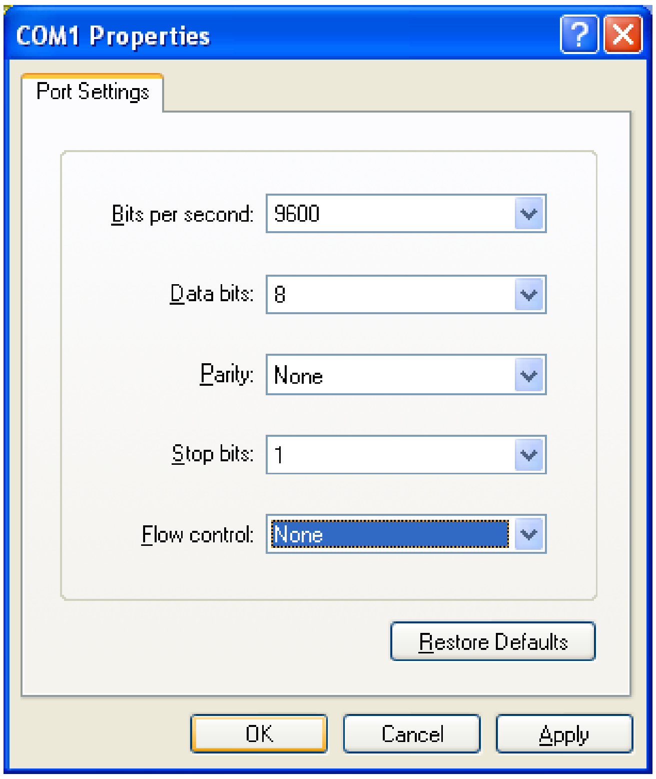
Phone number:

Connect using: IBM Integrated 56K Modem

IBM Integrated 56K Modem
COM4
COM1
TCP/IP (Winsock)

3. Configure the communication port properties as shown in *Figure 10*. Windows HyperTerminal is launched and communications can start.

Figure 10. Configuring connection properties



COM1 Properties

Port Settings

Bits per second: 9600

Data bits: 8

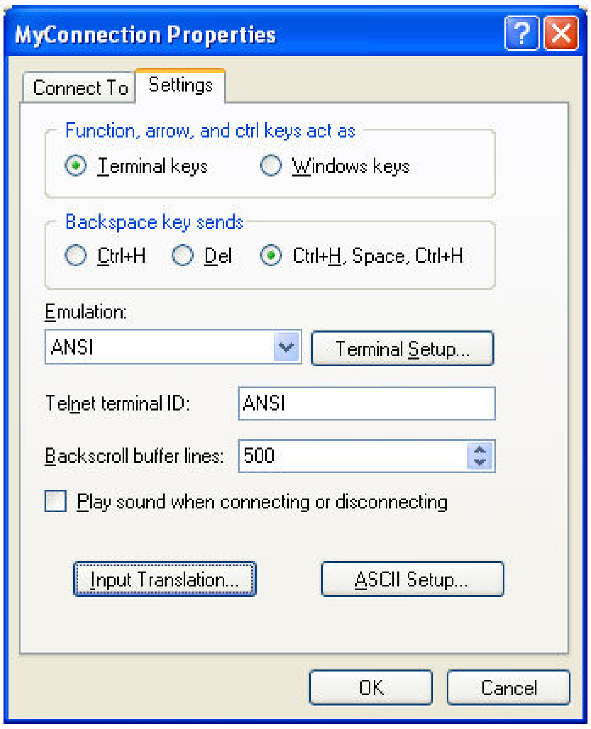
Parity: None

Stop bits: 1

Flow control: None

4. To check communication settings:
   a) Disconnect the HyperTerminal by choosing **Call > Disconnect** from the HyperTerminal main menu.
   b) Once communications are stopped, go to the **Settings** tab in **MyConnection Properties** menu. The parameters should be as shown below.

Figure 11. Checking communication settings



MyConnection Properties

Connect To | Settings

Function, arrow, and ctrl keys act as: [✓ Terminal keys] Windows keys

Backspace key sends: Ctrl+H Del [✓ Ctrl+H, Space, Ctrl+H]

Emulation: ANSI

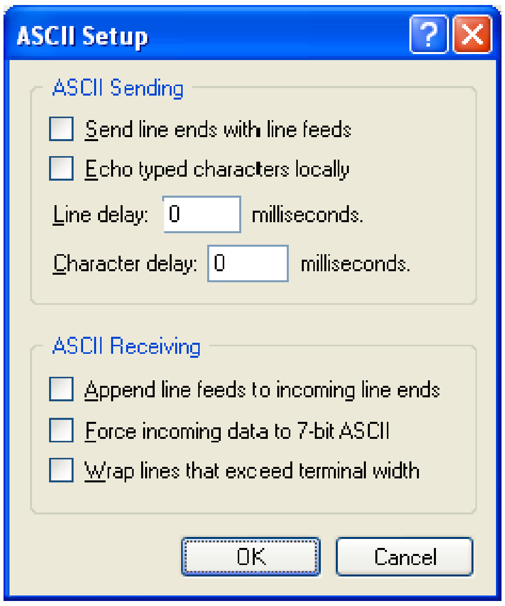
Telnet terminal ID: ANSI

Backscroll buffer lines: 500

☐ Play sound when connecting or disconnecting

c) Finally, click **ASCII Setup** in **MyConnection properties** menu, check that the ASCII parameters match those shown in *Figure 12*, and modify them if needed.

Figure 12. ASCII Setup parameters



ASCII Setup

ASCII Sending

- [ ] Send line ends with line feeds
- [ ] Echo typed characters locally

Line delay: 0 milliseconds.

Character delay: 0 milliseconds.

ASCII Receiving

- [ ] Append line feeds to incoming line ends
- [ ] Force incoming data to 7-bit ASCII
- [ ] Wrap lines that exceed terminal width

d) Close **MyConnection Properties** menu, and restart communications by choosing **Call** > **Call** from the HyperTerminal main menu. Your STM8S-DISCOVERY application is now ready to start.

Revision history

Table 4. Document revision history

Date	Revision	Changes
17-Dec-2010	1	Initial release.