AN5181

Application note

Building a thermometer using the STM8 Nucleo-64 boards

Introduction

The NUCLEO-8S208RB (built around the STM8S208RBT6 device) and the NUCLEO-8L152R8 (built around the STM8L152R8T6 device) are boards that allow the evaluation of the main features of all the STM8S Series and STM8L Series microcontrollers.

This application note demonstrates how to build a simple thermometer based on the STM8 Nucleo-64 boards and the LM235 precision temperature sensor. The STM8S208RBT6 or STM8L152R8T6 microcontroller (depending on the corresponding board) reads the temperature values and transmits them through the UART interface. The temperature values are then displayed on a terminal window (possibly based on a Windows HyperTerminal) of a PC connected to the UART through an RS232 or FTDI cable.

Once the STM8 Nucleo-64 has been powered-up through a USB cable connected to the host PC, an informative message is displayed on the terminal window and the user is prompted to enter the 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 a one-hour period are recorded in the MCU data EEPROM once per hour. They can be displayed any time by pressing a push button.

Table 1. **Applicable products**

Type	Part number
Evaluation boards	NUCLEO-8S208RB
	NUCLEO-8L152R8

Reference documents

- Available from STMicroelectronics website:
  - *STM8 Nucleo-64 boards* data brief (DB3591)
  - *STM8L152R8T6 Nucleo-64 board* user manual (UM2351)
  - *STM8S208RBT6 Nucleo-64 board* user manual (UM2364)
  - *ST232C 5 V powered multi-channel RS-232 driver and receiver* datasheet (DS1588)
  - *Precision temperature sensors* datasheet (DS0437) for LM235

1 Prerequisites

The material required to run the STM8 Nucleo-64 boards thermometer demonstration application is the following:

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

2 Application description

2.1 Hardware requirements

This application uses the STM8 Nucleo-64 boards on-board LED (LD2) together with its associated resistor (R1). The external passive components required by the application are listed in the table below.

Table 2. **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
Button 1	-	Standard push-button
Button 2 (on-board user push-button)	-	Standard push-button
C2, C3	100 nF	Charge-pump capacitors
C1, C4	100 nF	Output capacitors
C5	100 nF	Decoupling capacitor

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 STM8 UART input/output operating at 5 V.

This component is available in an SO16 package which fits the STM8 Nucleo-64 boards footprint. Refer to the corresponding datasheet for more information on the ST232C. The table below lists the packaged components.

Table 3. **List of packaged components**

Part number	Component description	Package
ST232C (order code ST232CN)	Very-high speed ultra-low-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

2.2 Application schematics

The figures below shows how to interface the LM235 temperature sensor, the ST232C driver/receiver and the push-buttons with the STM8 Nucleo-64 boards.

Button1 and Button2 (on-board user push-button) require an RC debounce filter to avoid triggering several interrupts. The debounce filter for Button1 and Button2 (on-board user push-button) consist of C7 and C6 capacitors plus I/Os pin internal pull-up resistor (about 45 KΩ).

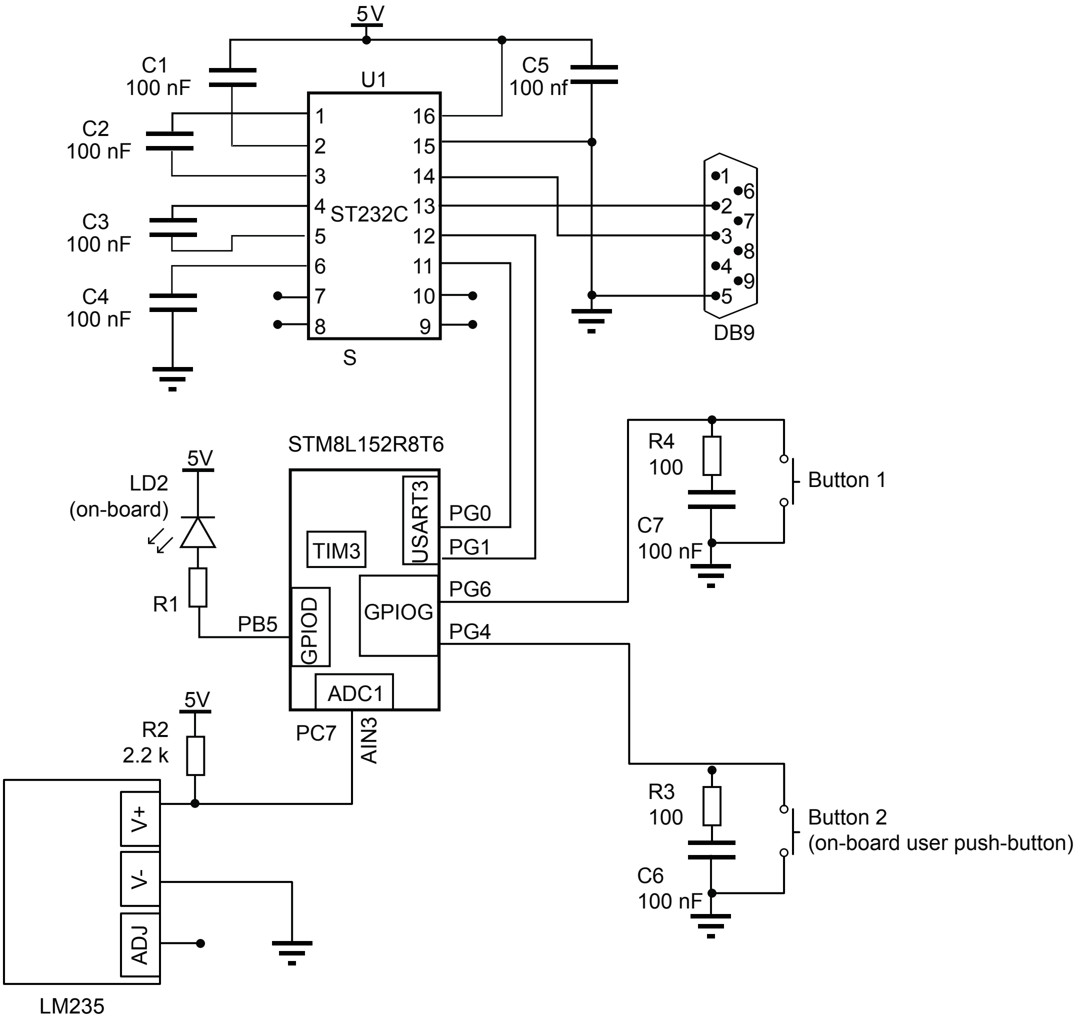
No external pull-up resistors are required as the internal pull-ups of the I/Os are used.

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 board schematics provided in the *STM8 Nucleo-64 boards* data brief (DB3591).

Figure 1. STM8L Series application schematic



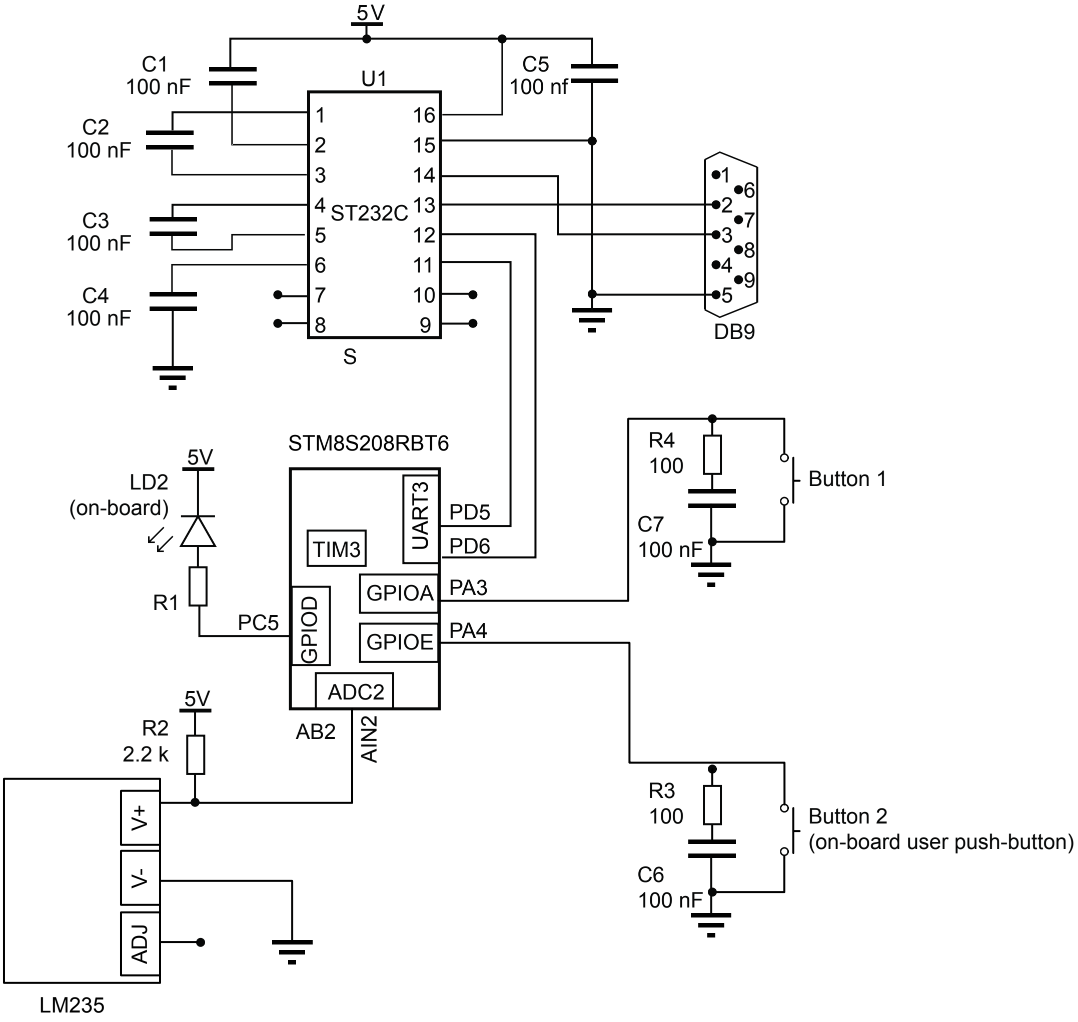


Figure 2. **STM8S Series application schematic**

2.3 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 ADC2 (for STM8S Series) or the ADC1 (for STM8L Series) every 50 ms at each timer interrupt. In order to improve the temperature measurement reliability, the temperature data is obtained by averaging the first 16 samples measured after 1 second has elapsed.

The resulting data is then compared to the current minimum and maximum temperature thresholds which can be modified if needed. LD2 is switched on if the temperature is below the low threshold or if it 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.

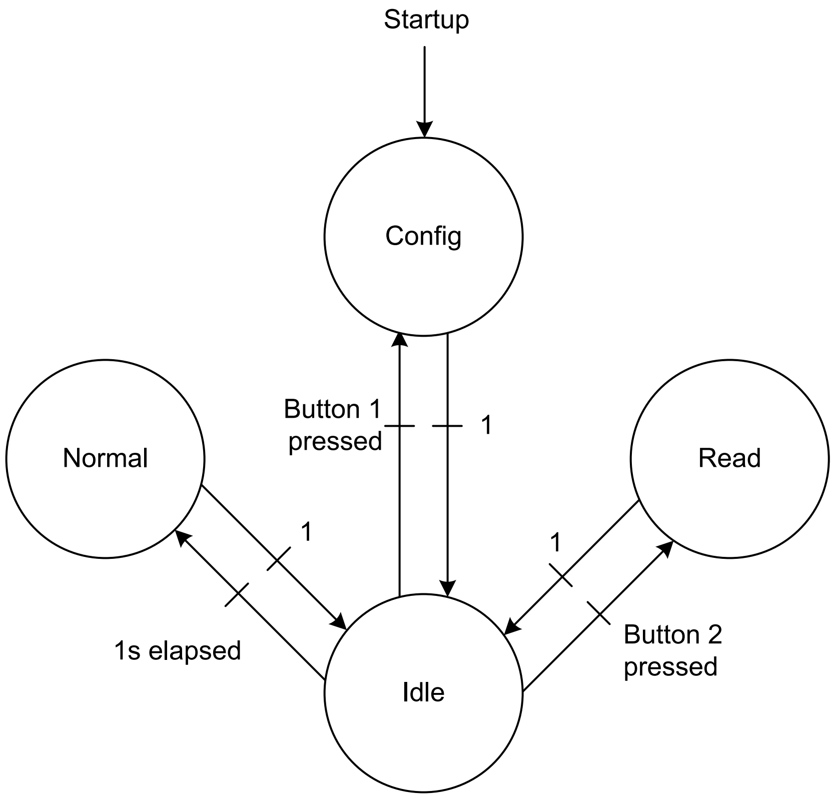
The minimum and maximum temperatures over one hour period are recorded in the data EEPROM once per hour and displayed on the terminal window.

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

Pressing Button2 (on-board user push-button) triggers the display on the terminal window of all recorded minimum and maximum temperatures stored in data EEPROM. LD2 is switched off.

The figure below shows the application state diagram.

Figure 3. **Application state machine**



The table below describes the actions performed by the application at each transition.

Table 4. **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 Button1 is pressed	Informative messages displayed on the terminal window. User prompted to enter minimum and maximum temperature thresholds for the first time or to update them.
State 2 Read	LD2 switched off	Button2 pressed	Minimum and maximum temperature values are read from data EEPROM and are displayed on the terminal window.

Application state	LED state	Entry condition	Actions
State 3 Normal	LD2 switched on if the temperature is out of range	Every 1 s Every 1 min Every 1 hr	*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 needed). *Every 1 hr:* minimum and maximum temperature over the previous hour recorded in data EEPROM.

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 3.3.3 State machine flowchart).

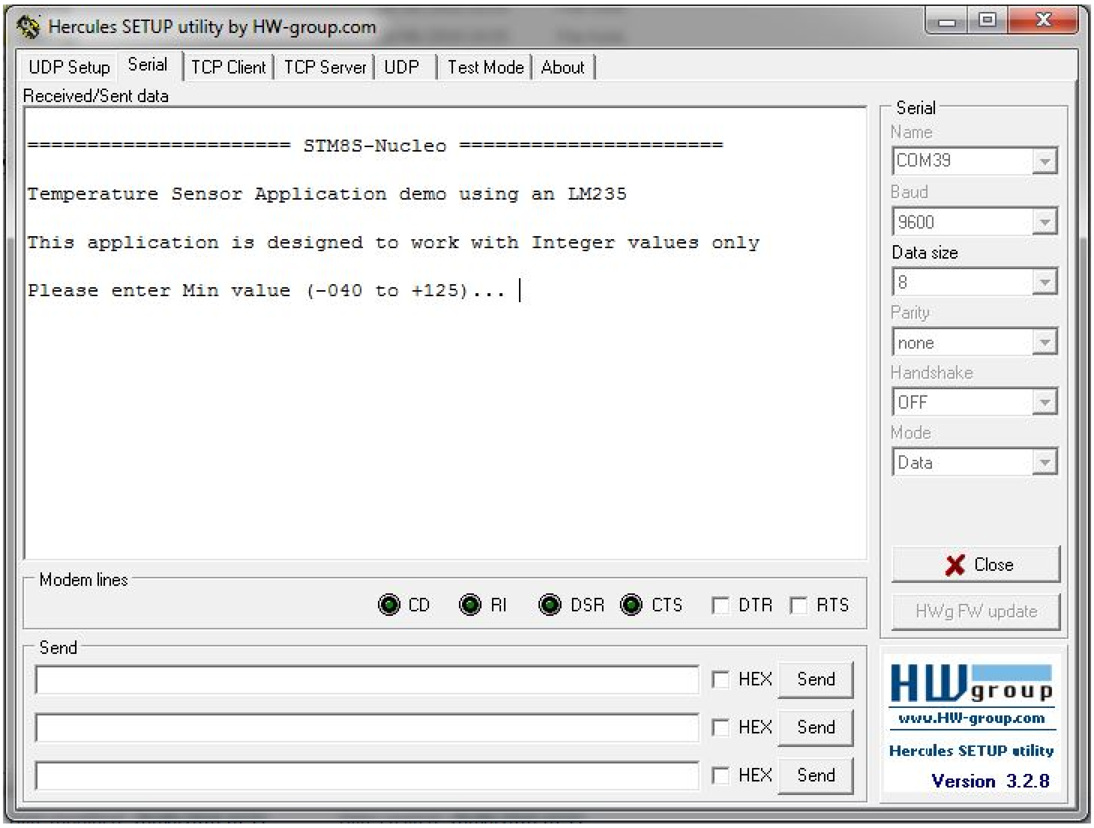
2.4 Launching the application

To display the terminal window, you can either run the preconfigured *thermometer.h* terminal based on Windows HyperTerminal (COM1 port) or create one by proceeding as explained in Section A Configuring the terminal window.

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.

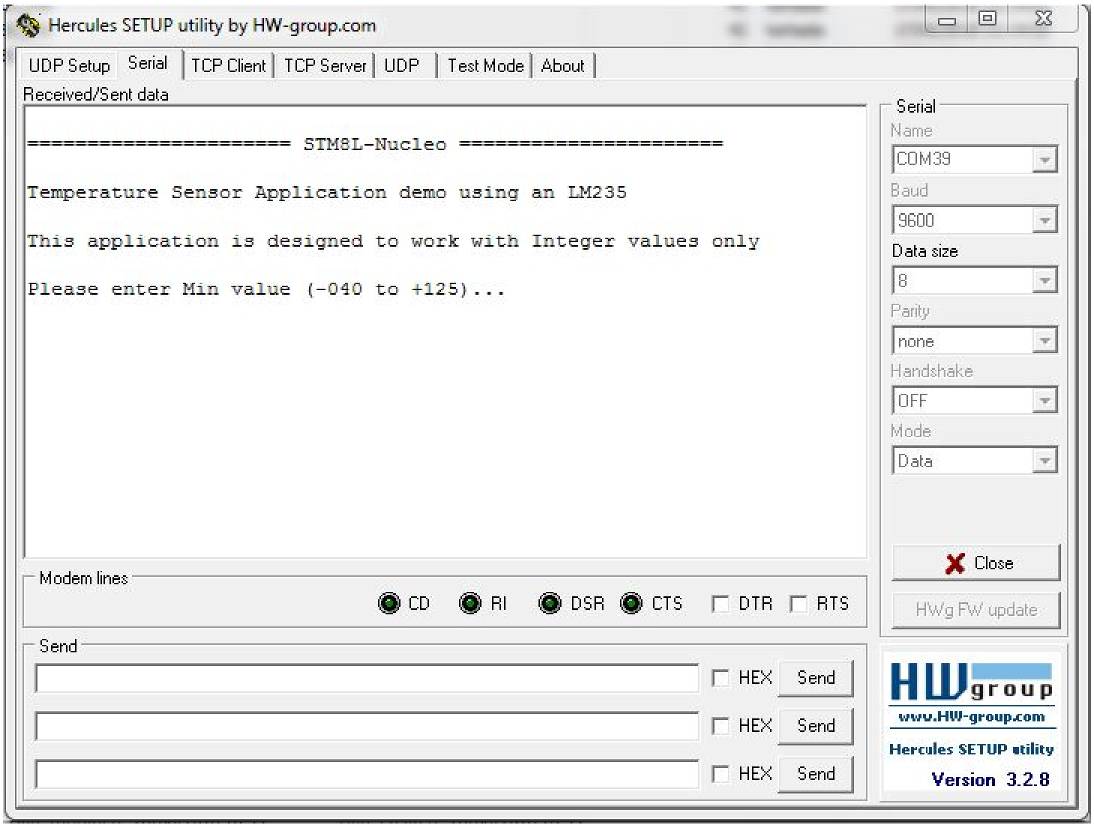
Figure 4. **STM8S terminal window at startup**



```
===================== STM8S-Nucleo =====================

Temperature Sensor Application demo using an LM235

This application is designed to work with Integer values only

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

Serial
Name: COM39
Baud: 9600
Data size: 8
Parity: none
Handshake: OFF
Mode: Data

Figure 5. **STM8L terminal window at startup**



```
====================== STM8L-Nucleo ======================

Temperature Sensor Application demo using an LM235

This application is designed to work with Integer values only

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

Serial
Name: COM39
Baud: 9600
Data size: 8
Parity: none
Handshake: OFF
Mode: Data

3 Software description

3.1 STM8 peripherals used by the application

The thermometer application software does not use STM8S Series or STM8L Series standard firmware library. It rather consists in an optimized code by using direct register accesses to control and use the STM8S Series and STM8L Series 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 HSI clock as the 16 MHz master clock source and the CPU clock prescaler division factor to 1.
- **GPIOs**
  The STM8S Series and STM8L Series GPIOs are used to switch on and off LD2, and to interface with pushbuttons using the following configuration:
  - STM8S Series: PA3 and PE4 (only PE4 is on-board)
  - STM8L Series: PG6 and PG4 (only PG4 is on-board)
- **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 for STM8S Series or on PG6 or PG4 for STM8L Series.
- **Flash memory**
  The minimum and maximum temperature values over a one-hour period are saved in the data EEPROM for further display on the terminal window.
- **U(S)ART**
  STM8 Series uses UART3 and STM8L Series uses USART3 to communicate with the PC temrinal software. Its configuration is the one below:
  - Baud rate = 9600 baud
  - Word length = 8 bits
  - One-stop bit
  - No parity
  - Receive and transmit enabled

Note: For STM8L Series, the USART3 CLK must be disabled.

  Communications are managed by polling each receive and transmit operation on the UART3 (STM8S Series) or USART3 (STM8L Series) peripheral .
- **ADC**
  The channel 2 of the ADC2 for STM8S Series or the channel 3 of the ADC1 for STM8L Series 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.

3.2 Exclusion of the STM8S Series and STM8L Series standard firmware library

As this application uses optimized code, the*stm8s.h* and *stm8l.h* files must be modified not include the STM8S Series standard firmware library and STM8L Series standard firmware library respectively. This is done by commenting the following define statement:

```
#define USE_STDPERIPH_DRIVER
```

3.3 Application software flowcharts

This section gives an overview of the application software main loop, the interrupt function flowchart and the state machine flowchart as well as some reference to the terminal communication functions.

3.3.1 Main loop flowchart

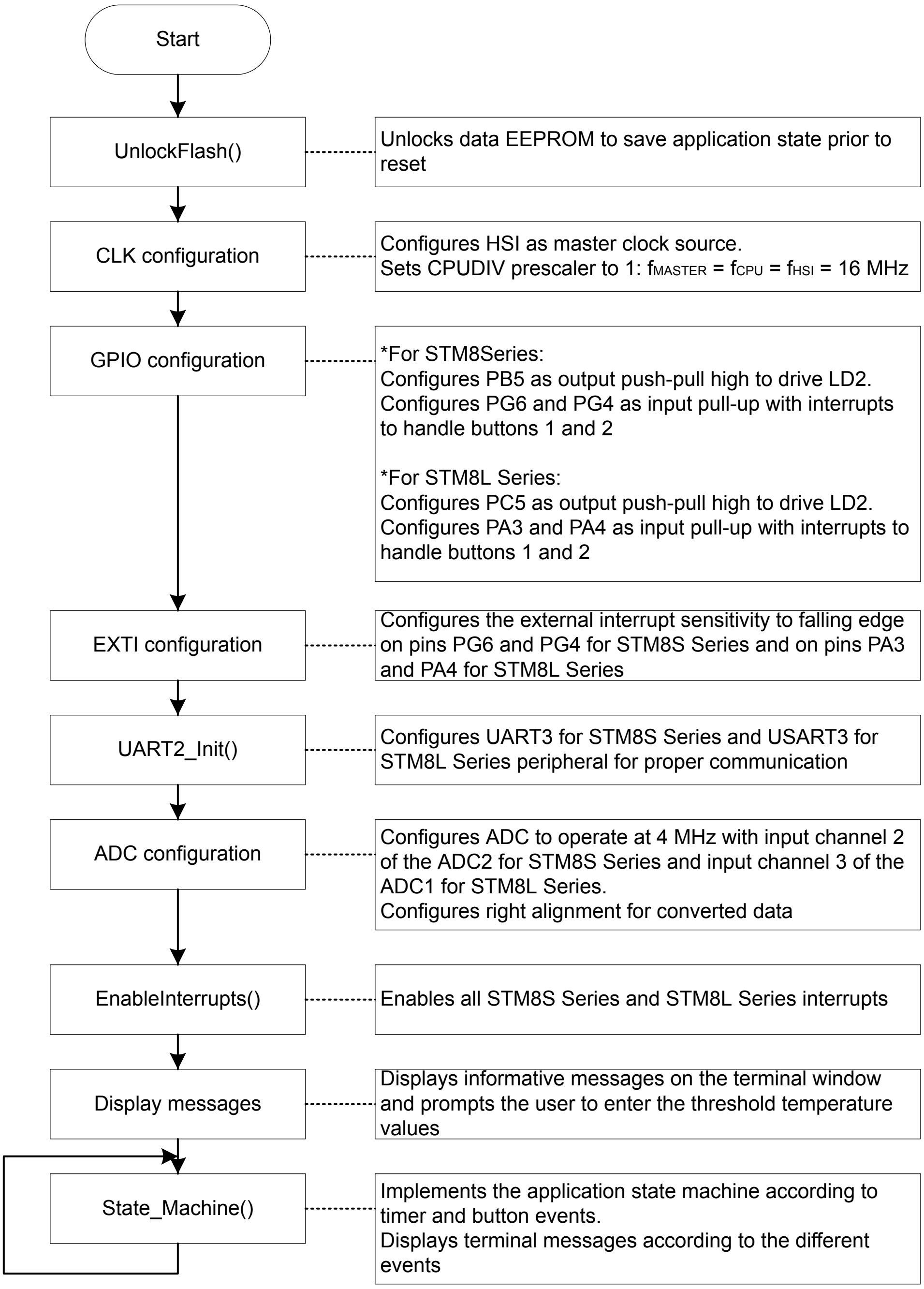
The main loop code initializes the required features, unlocks data EEPROM programming and calls the functions required to implement the general application algorithm.

To minimize the time drift of the software real time clock (refer to Timer-triggered acquisition), the HSI 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 enters the configuration mode directly (state = 1, see Section 3.3.3 State machine flowchart).

The figure below shows the flowchart of the application software main loop.

Figure 6. Main loop flowchart


Start
UnlockFlash()
Unlocks data EEPROM to save application state prior to reset
CLK configuration
Configures HSI as master clock source.
Sets CPUDIV prescaler to 1: fMASTER = fCPU = fHSI = 16 MHz
GPIO configuration
*For STM8Series:
Configures PB5 as output push-pull high to drive LD2.
Configures PG6 and PG4 as input pull-up with interrupts to handle buttons 1 and 2
*For STM8L Series:
Configures PC5 as output push-pull high to drive LD2.
Configures PA3 and PA4 as input pull-up with interrupts to handle buttons 1 and 2
EXTI configuration
Configures the external interrupt sensitivity to falling edge on pins PG6 and PG4 for STM8S Series and on pins PA3 and PA4 for STM8L Series
UART2_Init()
Configures UART3 for STM8S Series and USART3 for STM8L Series peripheral for proper communication
ADC configuration
Configures ADC to operate at 4 MHz with input channel 2 of the ADC2 for STM8S Series and input channel 3 of the ADC1 for STM8L Series.
Configures right alignment for converted data
EnableInterrupts()
Enables all STM8S Series and STM8L Series 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

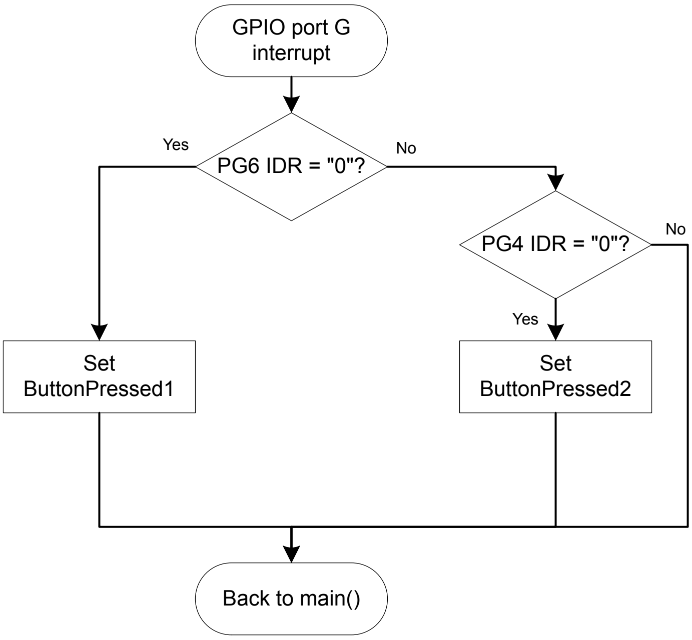
3.3.2 Interrupt function flowchart

Push button acquisition for STM8L Series

Each time Button1 or Button2 (on-board user push-button) is pressed, an interrupt is triggered and the PORTG_IRQhandler() function is called. The PORTG_IRQhandler() routine identifies which push button has been pressed by testing port G input register and asserts the ButtonPressed1 or ButtonPressed2 flag accordingly (see Section 3.3.3 State machine flowchart). These flags trigger a change of state in the application state machine.

The figure below shows the flowchart of the PORTG_IRQhandler() function.

Figure 7. STM8L Series: PORTG_IRQhandler() function flowchart

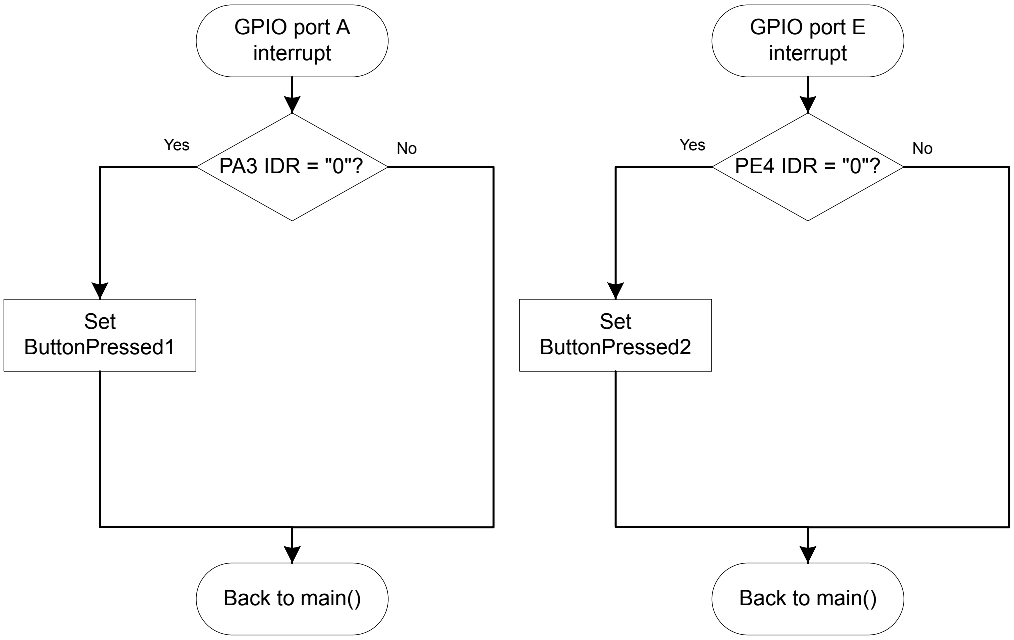


Push button acquisition for STM8S Series

Each time Button1 is pressed, an interrupt is triggered and the PORTA_IRQhandler() function is called. Each time Button2 (on-board user push-button) is pressed, an interrupt is triggered and the PORTE_IRQhandler() function is called. The routine asserts the ButtonPressed1 or ButtonPressed2 flag accordingly. These flags trigger a change of state in the application state machine.

The figure below shows the flowchaert of the PORTA_IRQhandler() and PORTE_IRQhandler() functions for STM8S Series.

Figure 8. STM8S Series: PORTA_IRQhandler() and PORTE_IRQhandler() functions flowcharts



Timer-triggered acquisition

TIMER3 is configured to generate an interrupt every 50 ms to trigger temperature acquisitions. After each conversion of ADC2 channel 2 (for STM8S Series) or ADC1 channel 3 (for STM8L Series), the digital value is stored in the buffer array for further computations and ADC 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 simulates 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 3.3.3 State machine flowchart).

The figure below shows the flowchart of the TIM3_IRQhandler() function.

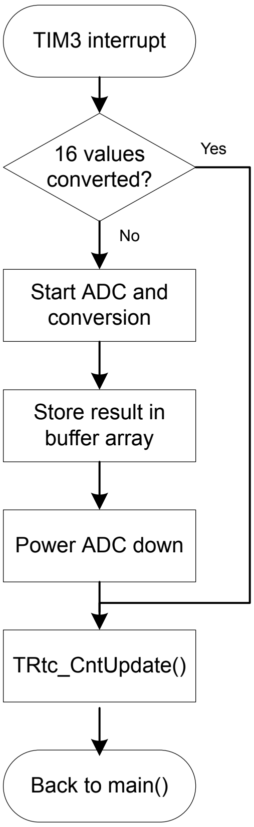


Figure 9. **TIM3_Init() function flowchart**

3.3.3 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 (on-board user push-button).
- **state 1 = Configuration 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. LD2 is also switched off. This state can onlye be entered by pressing Button2.
- **state 3 = Normal mode**

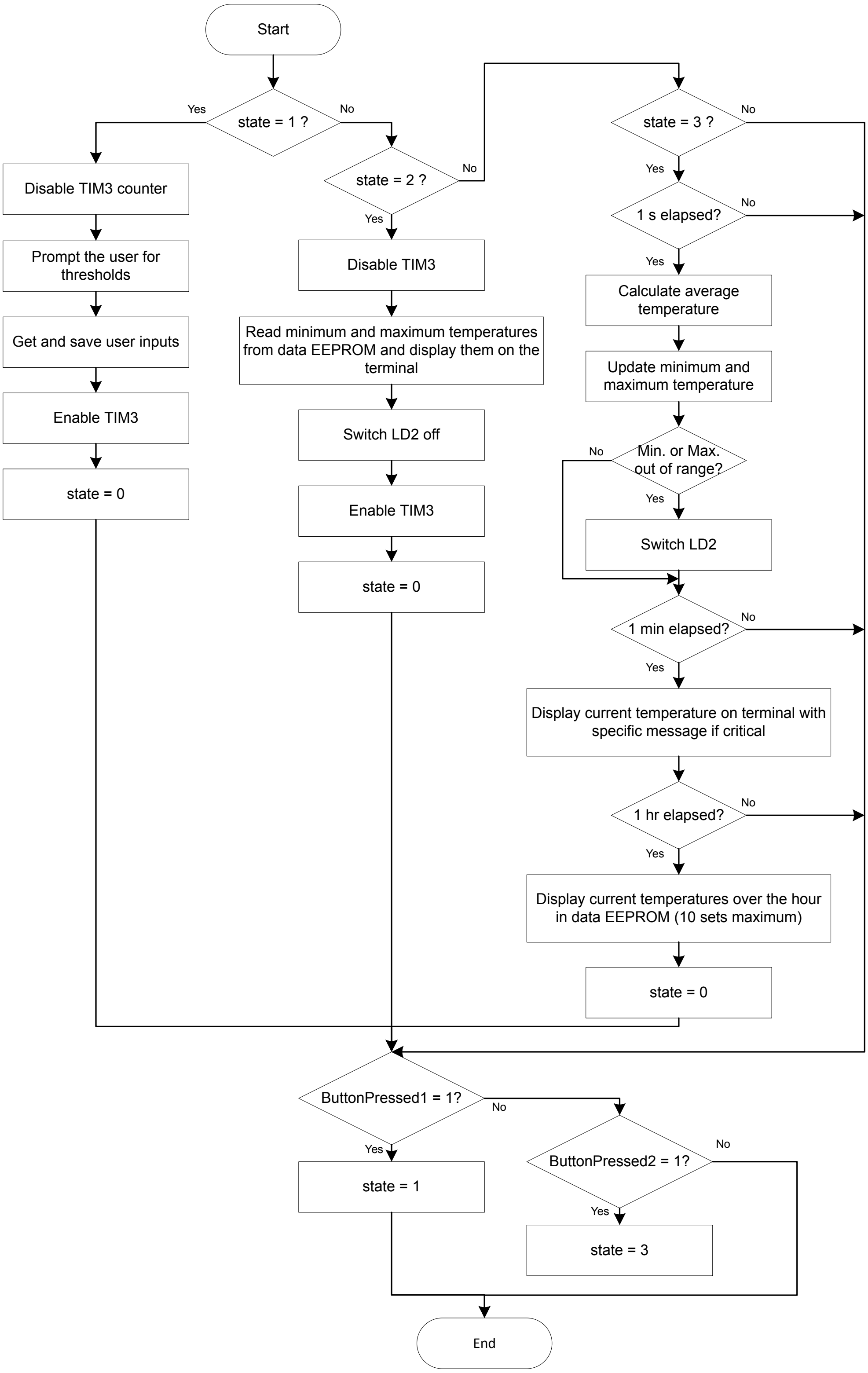
  The state machine Normal mode is entered every second. It is triggered by the TRtc_CntUpdate() routine that monitors the application time elapsed. Each time this state is entered, the average temperature is computed from 16 samples saved in the buffer array. If this temperature value exceeds the high threshold or is below the low threshold, LD2 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 is 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.

See below the flowchart of the State_Machine() function.

Figure 10. 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 ?
Yes
No
Disable TIM3
Read minimum and maximum temperatures from data EEPROM and display them on the terminal
Switch LD2 off
Enable TIM3
state = 0
state = 3 ?
Yes
No
1 s elapsed?
Yes
No
Calculate average temperature
Update minimum and maximum temperature
Min. or Max. out of range?
Yes
No
Switch LD2
1 min elapsed?
Yes
No
Display current temperature on terminal with specific message if critical
1 hr elapsed?
Yes
No
Display current temperatures over the hour in data EEPROM (10 sets maximum)
state = 0
ButtonPressed1 = 1?
Yes
No
state = 1
ButtonPressed2 = 1?
Yes
No
state = 3
End

3.3.4 Terminal communication functions

For a detailed description of the terminal communication functions, refer to Section A Configuring the terminal window.

A Configuring the terminal window

The terminal window connected to the STM8 Nucleo-64 board must be configured with the following settings valid for all terminal types:

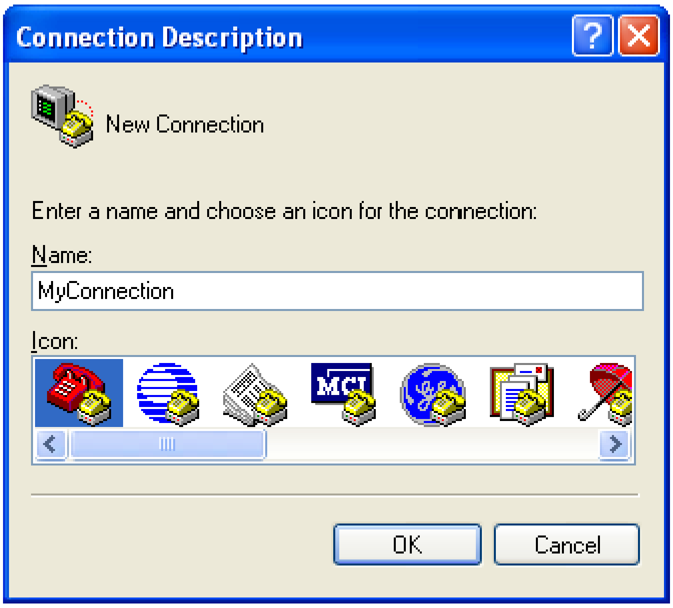
- Communication port: COM1 or other available
- Bits per second: 9600
- Data bits: 8
- Parity: none
- Stop bits: 1
- Flow control: none

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, the user can also set up a new connection with the STM8 Nucleo-64 board 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 11. Launch Windows HyperTerminal



Connection Description

New Connection

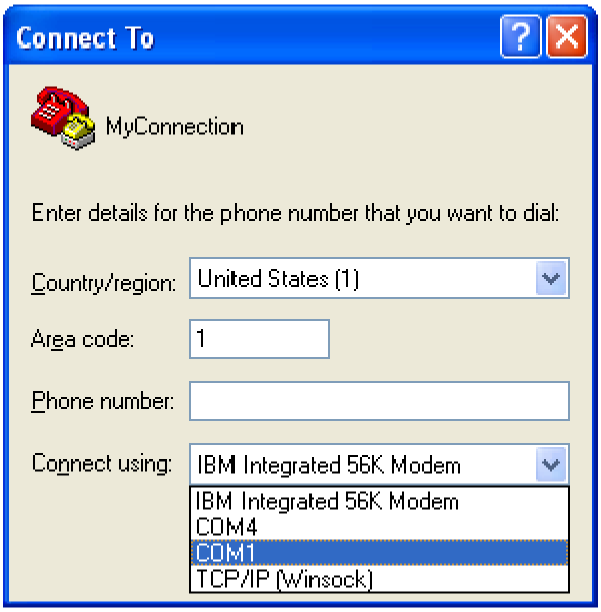
Enter a name and choose an icon for the connection:

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 12. **Select communication port**



Connect To

MyConnection

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

Country/region: United States (1)

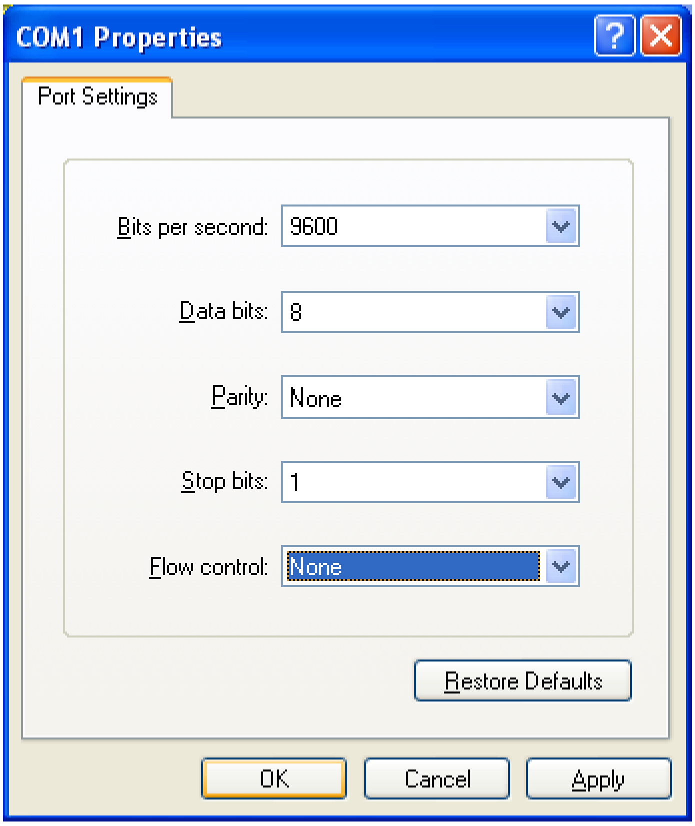
Area code: 1

Phone number:

Connect using: IBM Integrated 56K Modem

3. Configure the communication port properties as shown in the figure below. Windows HyperTerminal is launched and communications can start.

Figure 13. **Configure connection properties**



COM1 Properties

Port Settings

Bits per second: 9600

Data bits: 8

Parity: None

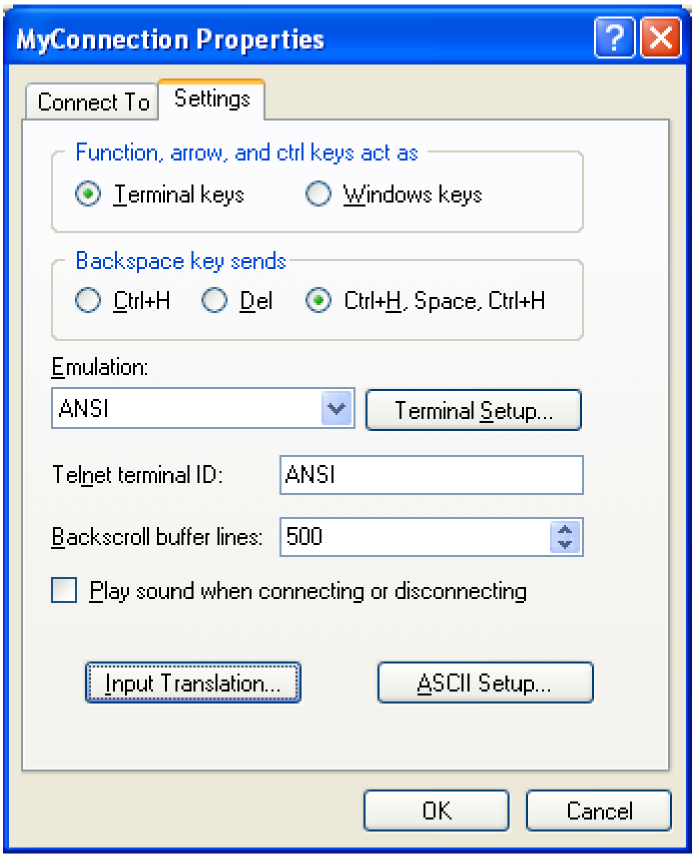
Stop bits: 1

Flow control: None

4. To check communication settings:
   – Disconnect the HyperTerminal by choosing **Call > Disconnect** from the HyperTerminal main menu.

– Once communications are stopped, go to the **Settings** tab in **MyConnection Properties** menu. The parameters should be set as shown below.

Figure 14. Check 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

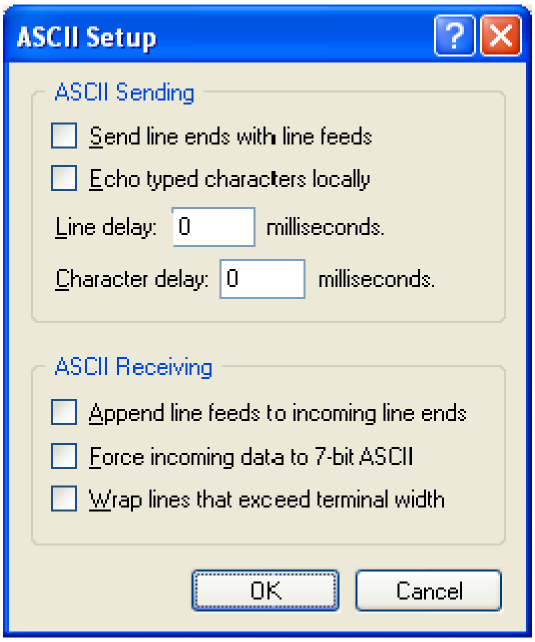
Telnet terminal ID: ANSI

Backscroll buffer lines: 500

□ Play sound when connecting or disconnecting

– Finally, click **ASCII Setup** in **MyConnection Properties** menu, and ensure that the ASCII parameters match those shown in the figure below.

Figure 15. **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

– Close **MyConnection Properties** menu, and restart communications by choosing **Call > Call** from the HyperTerminal main menu. The STM8 Nucleo-64 application is now ready to start.

Revision history

Table 5. Document revision history

Date	Version	Changes
29-Jun-2018	1	Initial release.

Contents

List of tables

List of figures