RS232 communications with a terminal using the STM8 Nucleo-64 boards

Introduction

This application note describes how to control the STM8 Nucleo-64 boards from a terminal window running on a PC that is connected to the UART of an STM8S208RBT6 (for NUCLEO-8S208RB) or STM8L152R8T6 (for NUCLEO-8L152R8) through an RS232 cable.

After adding the required components to the board and downloading the application software, the user is able to use a terminal to manage the STM8S Series or STM8L Series GPIOs and TIM3 timer, and to configure the beeper output.

Table 1. Applicable products

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

Reference documents

- *STM8 Nucleo-64 boards* data brief (DB3591)
- *STM8L152R8T6 Nucleo-64 board* user manual (UM2351)
- *STM8S208RBT6 Nucleo-64 board* user manual (UM2364)

1 Prerequisites

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

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

2 Configuring the NUCLEO-8S208RB board

Prior to run the application, the NUCLEO-8S208RB board (built around the STM8S208RBT6 device) must be configured to enable the beeper output. The beeper output is an STM8S208RBT6 alternate function. It is enabled by setting the alternate function remap option bit AFR7 in OPT2 option byte to ‘1’.

Note: The NUCLEO-8L152R8 board (built around the STM8L152R8T6 device) does not require that the user checks or activates the alternate function or the beeper.

3 Application description

3.1 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 description	Value
B1 buzzer	-
C1, C2, C3, C4, C5 capacitors	100 nF
DB9 connector	-

The application also uses of a 5 V powered ST232B RS232 driver/receiver (see the table below for more details). This extra component is essential since the COM port of the PC operates from a nominal 12 V power supply. This is not compatible with the STM8S Series or STM8L Series devices UART input/outputs operating at 5 V. This component is available in an SO16 package which fits the STM8 Nucleo-64 boards footprint. For more information on the ST232B refer to the ST232B datasheet.

Table 3. List of packaged components

Part name	Component description	Package
ST232B	Very-high speed ultra-low-power consumption 5 V RS232 driver/receiver used for UART 5/12 V level shifter	SO16

3.2 Application schematics

The figure below shows the application electrical schematics.

If the RS232 cable is not a null-modem cable (transmit and receive lines not crosslinked), connect U1 pin14 to DB9 pin2 and U1 pin13 to DB9 pin3.

Figure 1. STM8S Series application schematic

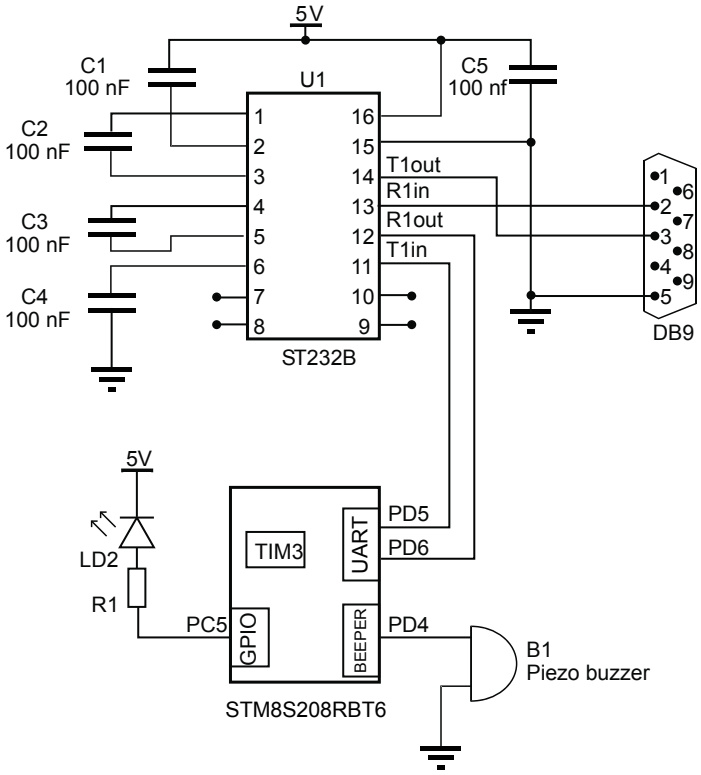
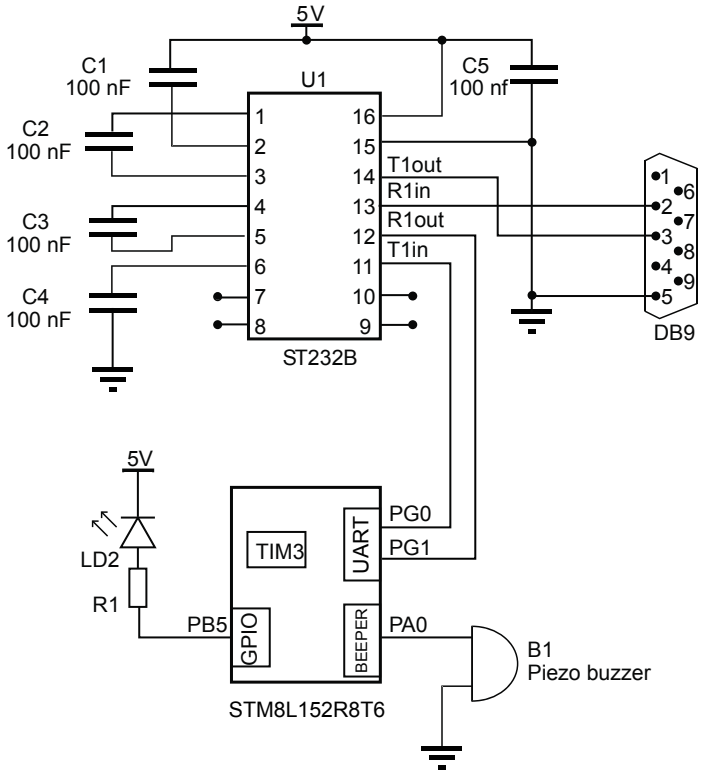


Figure 2. **STM8L Series application schematic**



3.3 Application principle

This application sets up a standard communication interface between the STM8S208RBT6 or STM8L152R8T6 microcontroller and a terminal window running on a PC. Communications are performed thanks to the STM8 devices UART using the RS232 protocol. Both the terminal window and the UART must be configured in the same way (see Section B Configuring the terminal window).

This document describes only the communications and data processing from the STM8 Nucleo-64 boards UART side. For more information about Windows HyperTerminal or similar software, refer to Microsoft® help or to the suppliers web pages.

3.3.1 Running the application

To run the application, perform the following steps:

1. Launch and configure a terminal window on your PC (see Section B Configuring the terminal window for an example using Windows HyperTerminal).
2. Compile and run the application firmware using the ST Visual Develop (STVD) or other toolchains.
3. Connect your PC to the STM8 Nucleo-64 through an RS232 cable.
4. When the application has started, a menu is displayed on the Windows HyperTerminal. This menu allows the user to:
   - Switch LD2 on or off.
   - Activate LD2 in Blinking mode.
   - Enable/disable the beeper and select the beep frequency.

   All the information displayed on this menu is sent by the STM8S Series or STM8L Series microcontroller. When a key is struck on the HyperTerminal, the corresponding ASCII value is sent to the microcontroller and decoded.

3.3.2 Communication sequence between the STM8 Nucleo-64 boards and the terminal

1. The STM8S Series or STM8L Series microcontroller sends the character string "Enter your choice" to the PC terminal emulator software.
2. The terminal displays the string "Enter your choice".
3. The user strikes key "2" on his keyboard.
4. The terminal emulator software sends back the corresponding ASCII code (0x52) to the microcontroller (see Section A Standard ASCII character codes).
5. The microcontroller decodes the data received, sends back the code 0x52 for it to be displayed on the terminal, and stores the value "2" in memory.
6. The terminal emulator software receives the code 0x52 and displays "2".
7. The user strikes the "Return" key.
8. The terminal emulator software send back the code 0x0D corresponding to carriage return (see Section A Standard ASCII character codes).
9. The STM8S Series or STM8L Series microcontroller decodes the received data, sends back the code 0x0D for it to be displayed it on the terminal, and performs the action associated to option 2.

4 Software description

4.1 STM8S Series and STM8L Series peripherals used by the application

This application example uses the STM8S Series and STM8L Series standard firmware library to control the general purpose functions. This application uses the following peripherals:

- **UART3 for STM8S Series or USART3 for STM8L Series:** it is used to communicate with the terminal window running on the PC. It must be configured as follows:
  - Baud rate = 9600 baud
  - Word length = 8 bits
  - One stop bit
  - No parity
  - Receive and transmit enabled

Note: If the STM8L Series are used, the USART3 clk must be disabled.

The communications are managed by polling each receive and transmit operation.

Note: The terminal window and the STM8 device UART peripheral must be configured with the same baud rate, word length, number of stop bits, and parity.

- **Timer3 (TIM3):** TIM3 timer is configured as a timebase with interrupt enabled to control LD2 blinking speed.
- **GPIOS:** the GPIOs are used to interface the MCU with the external hardware. Port PC5 for STM8 Series or port PB5 for STM8L Series is configured as output push-pull low to drive LD2.
- **BEEPER:** to drive the buzzer, the BEEPER peripheral outputs a signal of 1, 2, or 4 KHz on the BEEP output pin.

4.2 STM8 standard firmware library configuration

4.2.1 STM8S Series standard firmware library

The *stm8s_conf.h* file of the STM8S Series standard firmware library allows to configure the library by enabling the peripheral functions used by the application.

The following define statements must be present:

- `#define _GPIO 1` enables the GPIOs
- `#define _TIM3 1` enables TIM3
- `#define _BEEPER 1` enables the BEEPER
- `#define _UART3 1` enables UART3

4.2.2 STM8L Series standard firmware library

The *stm8l_conf.h* file of the STM8L Series standard firmware library allows to configure the library by enabling the peripheral functions used by the application.

The following define-statements must be present:

- `#include "stm8l15x_gpio.h"`
- `#include "stm8l15x_tim2.h"`
- `#include "stm8l15x_tim3.h"`
- `#include "stm8l15x_beep.h"`
- `#include "stm8l15x_usart.h"`

4.3 Application software flowcharts

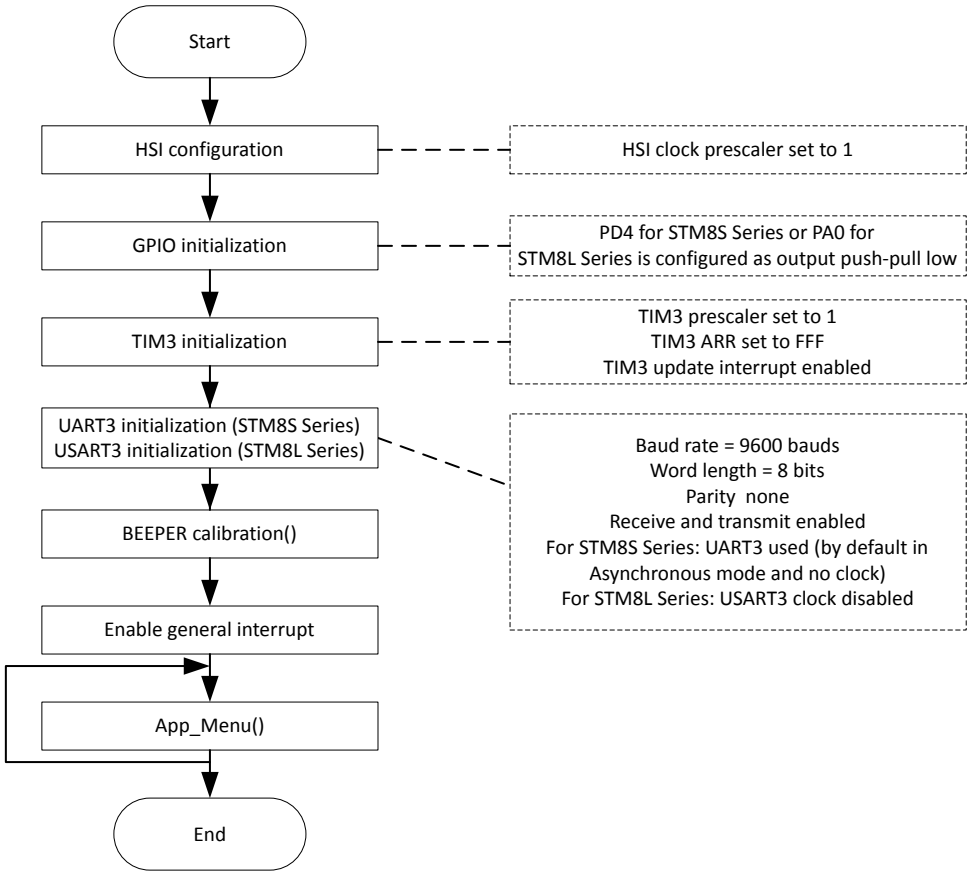
This section describes the application software main loop and the function that controls data reception/ transmission from/to the terminal window:

- **App_Menu**
  This function is used to display a menu on the terminal, and manage the information entered by the user.
- **SerialPutString**
  This function is used to transmit a string to the terminal.
- **SerialPutChar**
  This function is used to transmit a character to the terminal.
- **GetInputString**
  This function is used to receive a string from the terminal.
- **GetIntegerInput**
  This function is used to receive an integer from the terminal.
- **Get_Key**
  When a key is stroke, this function returns the corresponding hexadecimal code.

4.3.1 Application main routine

The application main routine configures the STM8S Series or STM8L Series peripherals and enables all the standard interrupts used by the application. When the initialization is complete, the main routine displays the application menu on the terminal window.

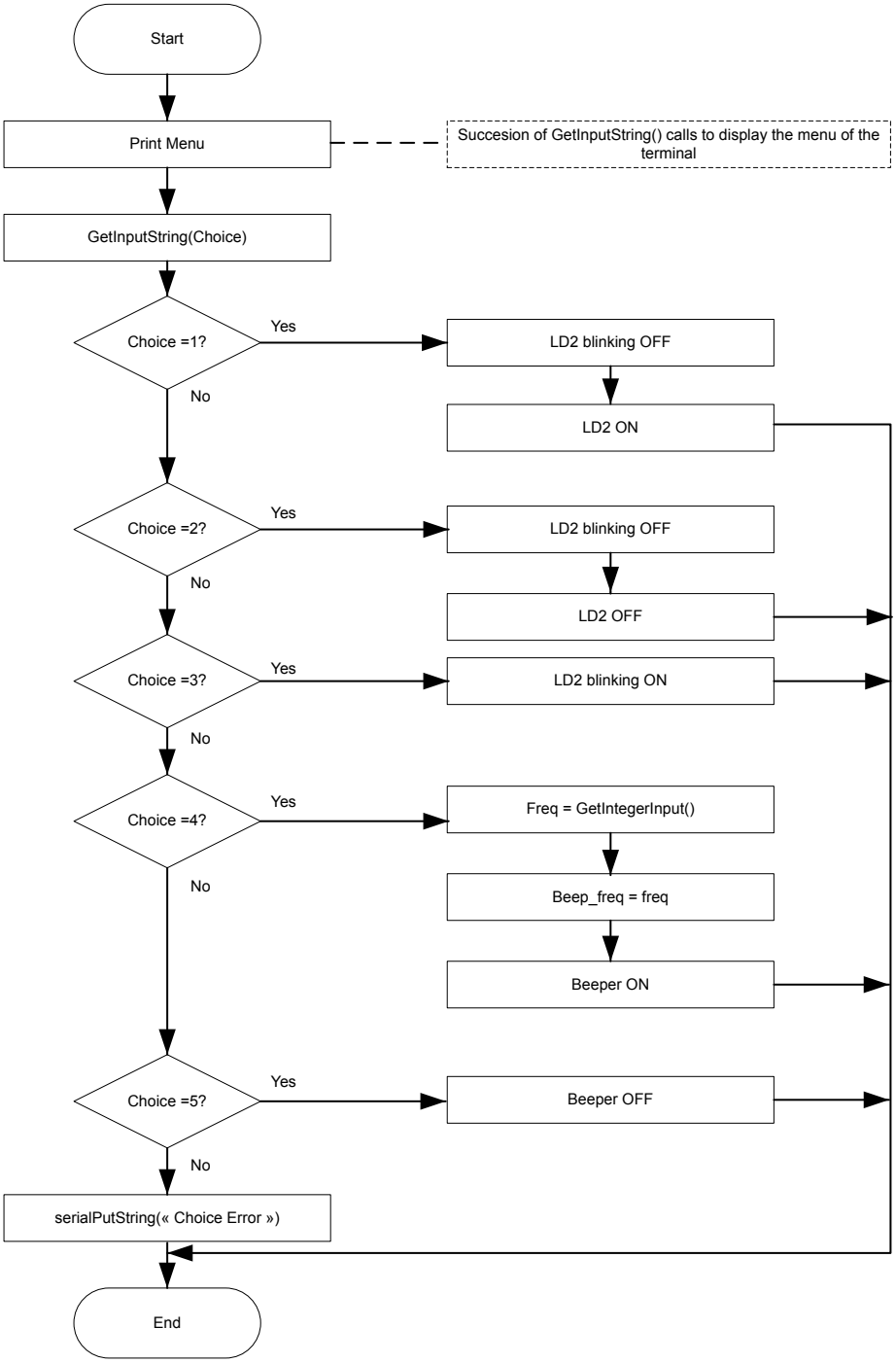
Figure 3. Main routine flowchart



4.3.2 App_menu function

The App_menu function is the main application routine. It displays a menu on the terminal through which the GPIOs, TIM3 and BEEPER can be configured. App_menu calls GetInputString, GetIntegerInput and SerialPutString to send and receive data through the RS232 interface.

Figure 4. App_menu flowchart



4.3.3 GetInputString function

The GetInputString function is used to receive and store the character strings sent through the terminal window. This function relies on the Get_key function to acquire and decode each character (see dedicated section). Different actions can be performed according to the value of the character ASCII code:

- If ASCII code = '\b'

  A backspace has been sent by the terminal. The last character of the string is erased if the string is not empty.
- If ASCII code belongs to {0...1 or a...Z}

  The character is stored.
- If ASCII code = '\r'

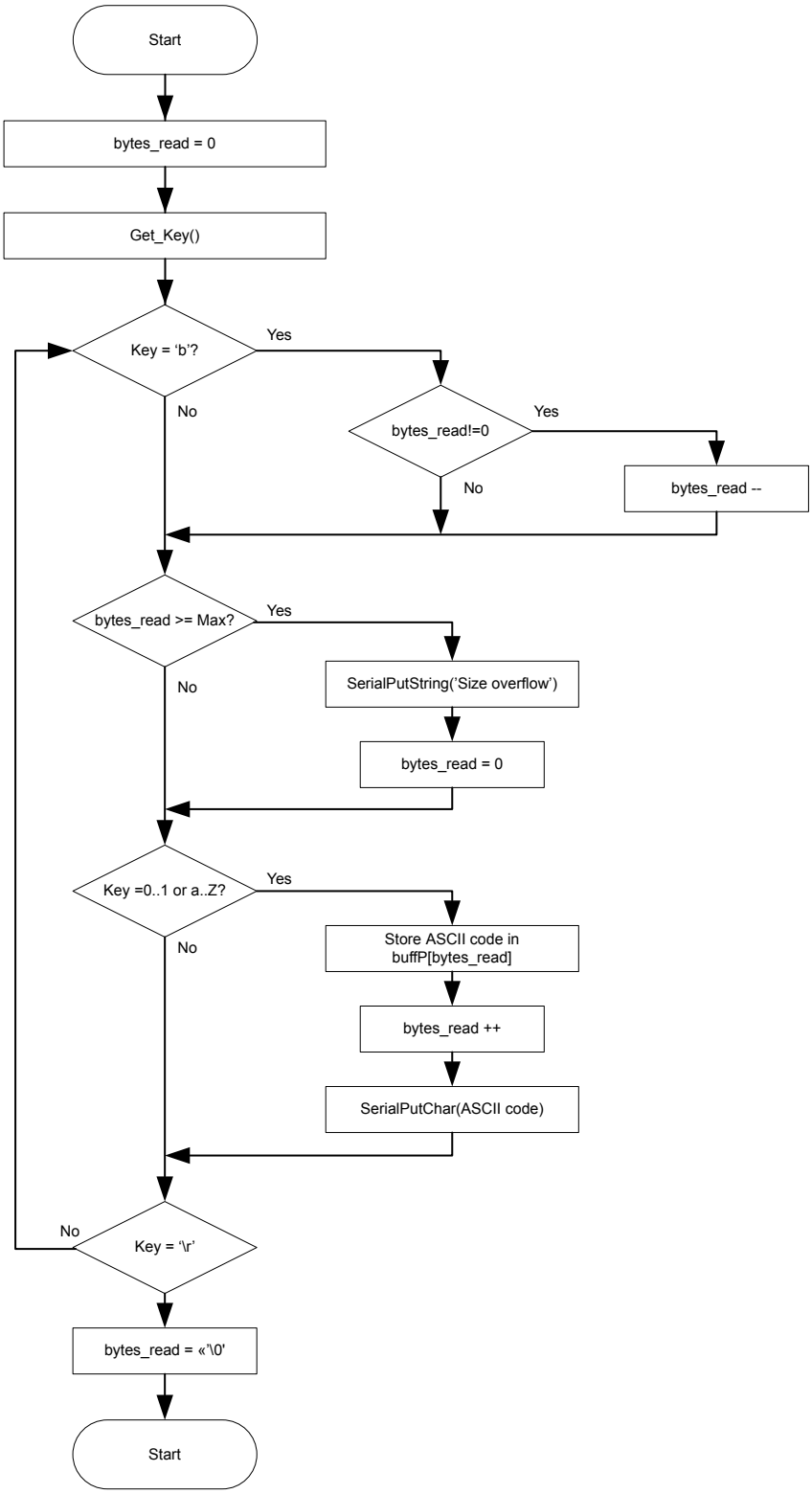
  The GetInputString function stores the "end of string" value, '\0', at the end of the string.

  The maximum number of ASCII codes stored in the buffP[bytes_read] buffer has been reached.

  The software erases the recorded string and waits for another input from the terminal.

For more information on ASCII codes refer to Section A Standard ASCII character codes.

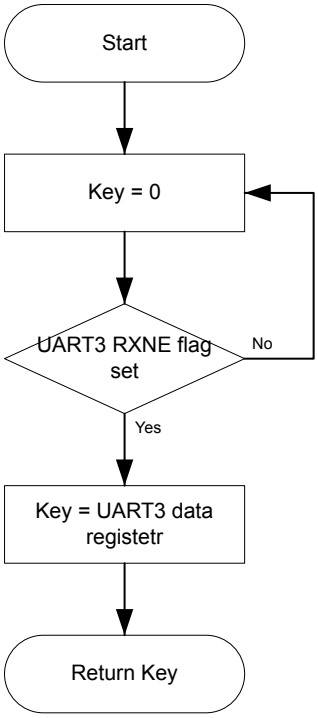
Figure 5. GetInputstring flowchart



4.3.4 Get_key function

The Get_key function is used to detect a key stroke on the terminal by polling the UART RXNE flag. This function returns the received value.

Figure 6. Get_key function flowchart



4.3.5 SerialPutString and SerialPutChar functions

The SerialPutString function is used to send a character string through the UART. The string characters are sent one by one by the SerialPutChar function as described in the flowcharts below.

Figure 7. **SerialPutString flowchart**

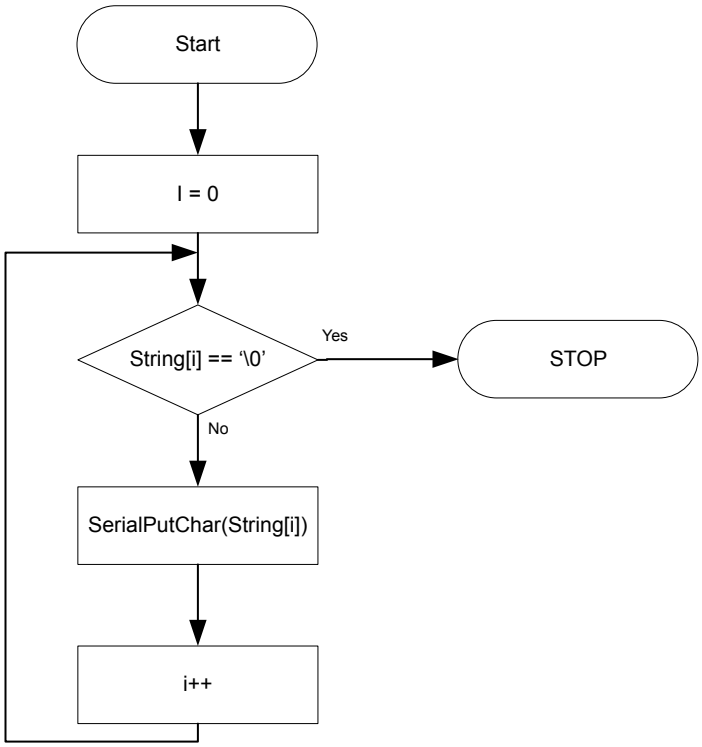
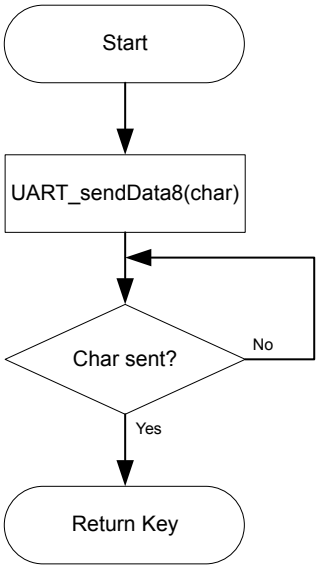


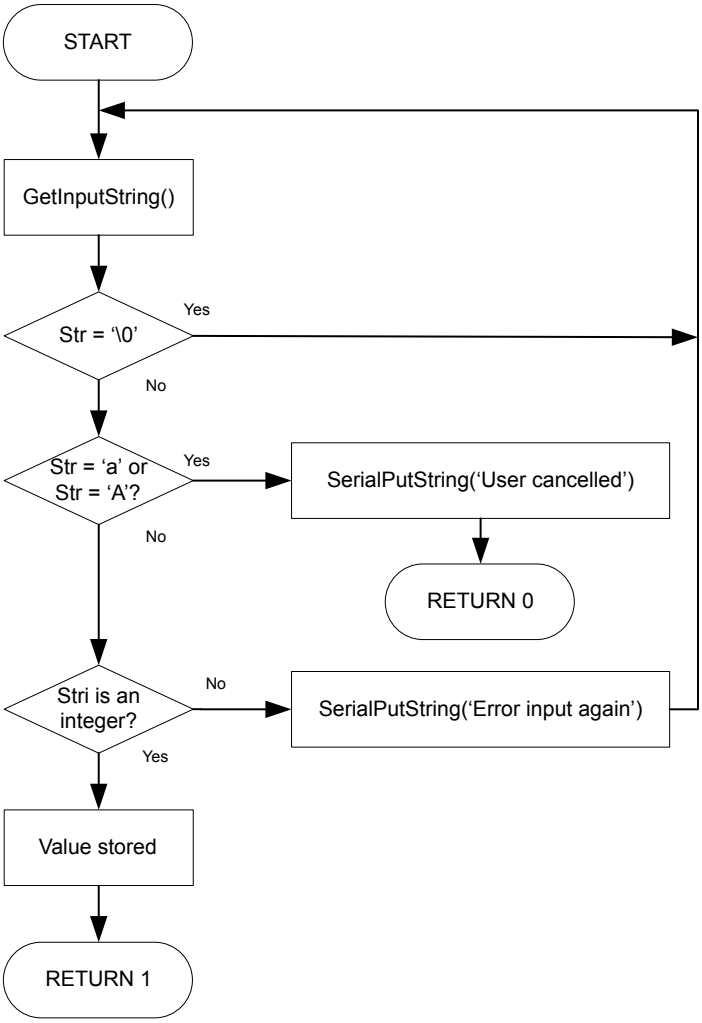
Figure 8. **SerialPutChar flowchart**



4.3.6 GetIntegerInput function

The GetIntegerInput function is used to check that incoming data correspond to an integer. If it does, the data is stored in the memory, otherwise the user is prompted to enter new data.

Figure 9. GetIntegerInput flowchart



A Standard ASCII character codes

Table 4. Standard ASCII character codes

Hex	Char	Hex	Char	Hex	Char	Hex	Char
0x00	NULL	0x20	Space	0x40	@	0x60	‘
0x01	Start of heading	0x21	!	0x41	A	0x61	a
0x02	Start of text	0x22	“	0x42	B	0x62	b
0x03	End of text	0x23	#	0x43	C	0x63	c
0x04	End of transmit	0x24	$	0x44	D	0x64	d
0x05	Enquiry	0x25	%	0x45	E	0x65	e
0x06	Ack	0x26	&	0x46	F	0x66	f
0x07	Audible bell	0x27	‘	0x47	G	0x67	g
0x08	Backspace	0x28	(	0x48	H	0x68	h
0x09	Horizontal tab	0x29	)	0x49	I	0x69	i
0x0A	line feed	0x2A	*	0x4A	J	0x6A	j
0x0B	Vertical tab	0x2B	+	0x4B	K	0x6B	k
0x0C	Form feed	0x2C	,	0x4C	L	0x6C	l
0x0D	carriage return	0x2D	-	0x4D	M	0x6D	m
0x0E	Shift out	0x2E	.	0x4E	N	0x6E	n
0x0F	Shift in	0x2F	/	0x5F	O	0x6F	o
0x10	Data link escape	0x30	0	0x50	P	0x70	p
0x11	Device control 1	0x31	1	0x51	Q	0x71	q
0x12	Device control 2	0x32	2	0x52	R	0x72	r
0x13	Device control 3	0x33	3	0x53	S	0x73	s
0x14	Device control 4	0x34	4	0x54	T	0x74	t
0x15	Neg. Ack	0x35	5	0x55	U	0x75	u
0x16	Synchronous idle	0x36	6	0x56	V	0x76	v
0x17	End trans. block	0x37	7	0x57	W	0x77	w
0x18	Cancel	0x38	8	0x58	X	0x78	x
0x19	End of medium	0x39	9	0x59	Y	0x79	y
0x1A	Substitution	0x3A	:	0x5A	Z	0x7A	z
0x1B	Escape	0x3B	;	0x5B	[	0x7B	{
0x1C	File sep.	0x3C	<	0x5C	\	0x7C	|
0x1D	Group sep.	0x3D	=	0x5D	]	0x7D	}
0x1E	Record sep.	0x3E	>	0x5E	^	0x7E	~
0x1F	Unit sep.	0x3F	?	0x5F	_	0x7F	<DEL>

B 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**.

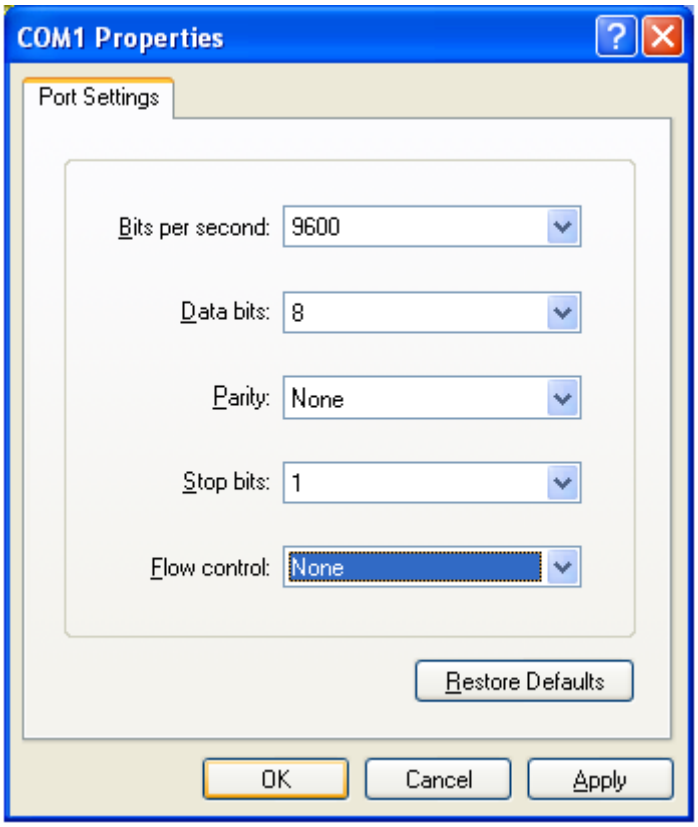
Figure 10. Launch Windows HyperTerminal

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 11. **Select communication port**

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

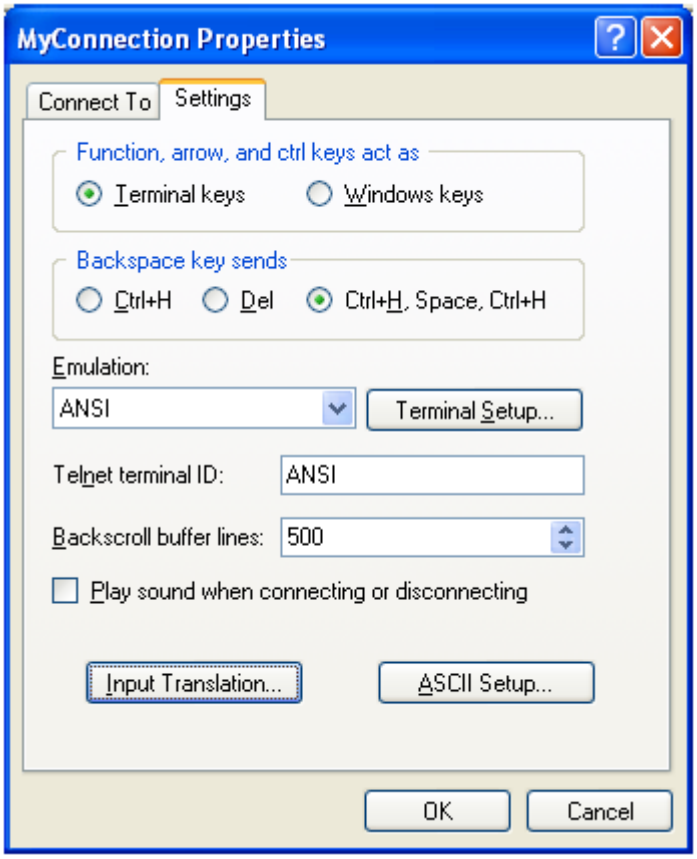
Figure 12. **Configure connection properties**



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

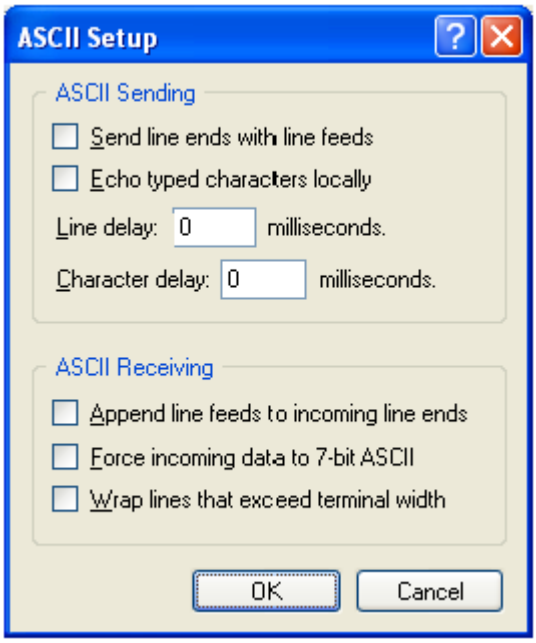
– Once communications are stopped, go to the **Settings** tab in **MyConnection Properties** menu. The parameters should be set as shown below.

Figure 13. Check communication settings



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

Figure 14. **ASCII setup parameters**



– 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