UM0884
User manual

RS232 communications with HyperTerminal using the STM8S-DISCOVERY

Application overview

This user manual provides a short description of how to interface a Windows HyperTerminal with STM8S microcontroller devices.

After adding the required components to the board and downloading the application software, you will be able to use an HyperTerminal to manage STM8S GPIOs and TIM3 timer, and to configure the beeper output.

Reference documents

- STM8S-DISCOVERY evaluation board use manual (UM0817).
- Developing and debugging your STM8S-DISCOVERY application code (UM0834).
- ST232B-ST232C datasheet

All these documents are available at http://www.st.com.

Contents

List of tables

List of figures

1 Application description

1.1 Hardware required

This application uses 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*.

The application also makes use of a 5 V ST232B RS232 driver/receiver (see *Table 2*). 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 UART input/outputs operating at 5 V. This component is available in an SO16 package which fits the STM8S-Discovery footprint. For more information on the ST232B refer to the ST232B datasheet.

An RS232 serial cable is also required to connect the HyperTerminal to the STM8S-DISCOVERY.

Table 1. List of passive components

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

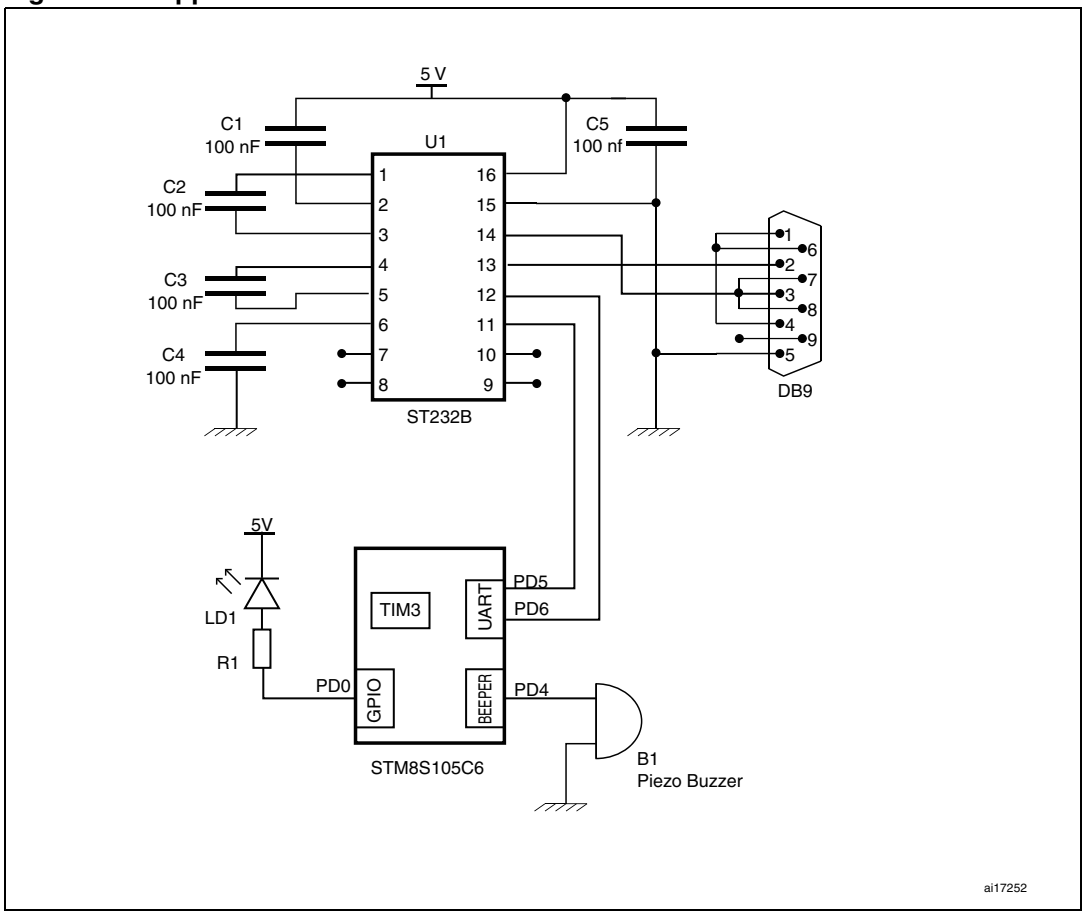
Table 2. List of packaged components

Part name	Component name	Description	Package
ST232B	Very-high speed ultralow-power consumption 5 V RS232 drivers and receivers	Level shifter 5/12 V for UART purpose.	SO16

1.2 Application schematics

Figure 1 shows the application electrical schematics.

Figure 1. Application schematics

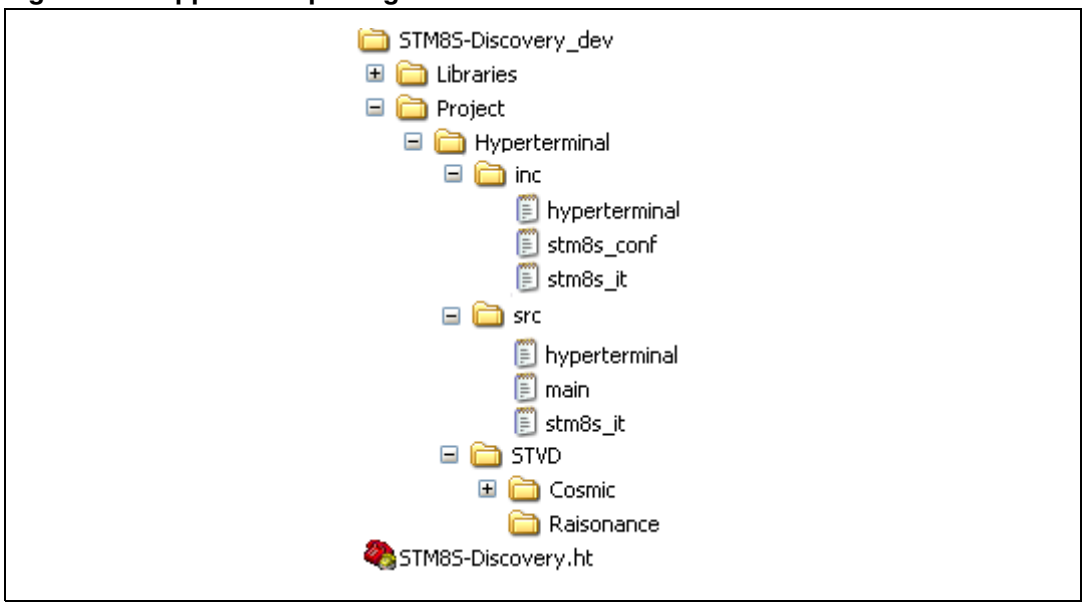


1.3 Description of the application package

All the functions allowing to easily embed the UART HyperTerminal capability into an application using an HyperTerminal, are provided within a HyperTerminal driver (.h and .c files) located in the *sources* and *includes* directories of the application package (see *Figure 2*).

An example of Microsoft® HyperTerminal configuration file (*STM8S-Discovery.ht*) is provided to automatically configure the baud rate and the other communication parameters from your PC. If communication problems occur, refer to *Appendix B: HyperTerminal configuration* for how to manually configure your Microsoft® HyperTerminal.

Figure 2. Application package architecture



1.4 Application principle

This application implements a standard interface between an STM8S microcontroller and a Windows HyperTerminal. Communications are performed through the STM8S UART peripheral and the serial PC port using the RS 232 protocol. The transmitter and receiver must be configured in the same way (see).

This document only describes the communications and data processing from the STM8S UART side. For more information about Windows HyperTerminal or similar software, refer to Microsoft® Help or suppliers web pages.

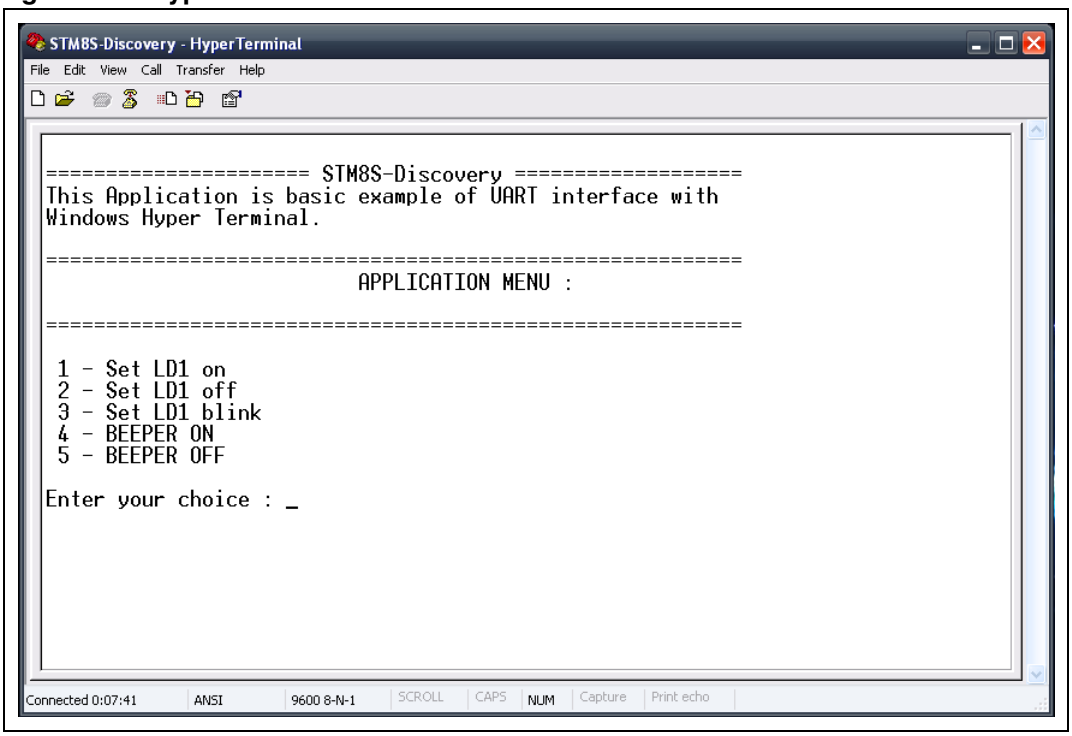
1.4.1 Running the HyperTerminal application

To run the HyperTerminal application, perform the following steps:

1. Run and configure Microsoft HyperTerminal on your PC (see).
2. Compile and run the application using the ST Visual Develop (STVD).
3. When the application has started, a menu is displayed on the Windows HyperTerminal (*Figure 3.: HyperTerminal menu*). It allows to:
   - Switch LD1 on or off through the PD0 port of the STM8S microcontroller.
   - Configure LD1 blinking speed.
   - Configure the beeper peripheral (peripheral switched on/off and beep frequency)

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

Figure 3. HyperTerminal menu



1.4.2 Communication principle when selecting a menu option using the HyperTerminal software

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

2 Software description

2.1 STM8S peripherals used by the application

This application example uses the STM8S standard firmware library to control general purpose functions. It makes use of the following STM8S peripherals:

UART2

UART2 is used to communicate with the PC HyperTerminal software. It must be configured as follows:

- Baud rate = 9600 baud
- Word length = 8 bits
- One stop bit
- Odd parity
- Receive and transmit enabled
- UART2 clock disabled

The communications are managed by polling each receive and transmit operation on UART2.

Note: The PC HyperTerminal software and the STM8S UART peripheral must be configured with the same baud rate, word length, number of stop bits, and parity. This is done through the hyperterminal.ht file.

TIM3

TIM3 timer is used to drive LD1 blinking speed. The blinking frequency is managed through a timer interruption.

GPIOs

The GPIOs are used to switch on and off LD1. The LED is driven by PD0 port configured in output push-pull low mode.

BEEPER

The BEEPER is used to drive a buzzer. A signal with a frequency ranging from 1.2 to 4 KHz is output on the Beeper pin.

Additional explanations on how to set the LED and the buzzer on and off are provided in the following user manuals:

- Adjustable LED blinking speed using STM8S-DISCOVERY touch sensing key (UM0833)
- Adjustable buzzer frequency using STM8S-DISCOVERY touch sensing key (UM0845)

2.2 Configuring STM8S standard firmware library

The *stm8s_conf.h* file of the STM8S 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 _UART2 1` enables UART2

2.3 Application software flowcharts

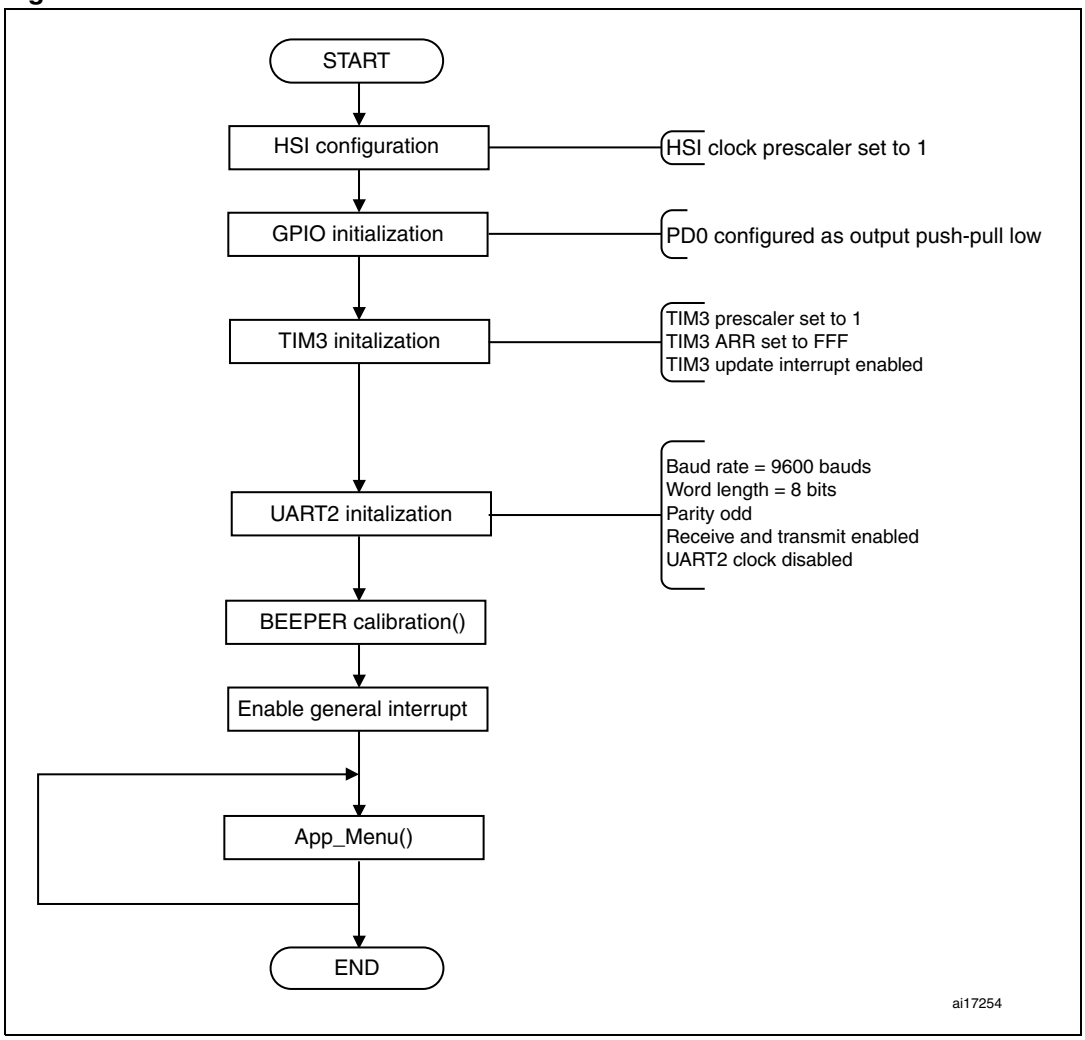
This section describes the main function together with all the Receive/Transmit functions contained in the STM8S HyperTerminal driver:

- **App_Menu()**
  This function is used to display a menu on the HyperTerminal, and manage the information entered by the user.
- **SerialPutString()**
  This function is used to transmit a string to the HyperTerminal.
- **SerialPutChar()**
  This function is used to transmit a character to the HyperTerminal.
- **SerialGetString()**
  This function is used to receive a string from the HyperTerminal.
- **SerialGetInteger()**
  This function is used to receive and Integer from the HyperTerminal. It calls the SerialGetString() function.
- **Get_Key()**
  This function is used by the SerialGetString() function to wait for a key stroke on the HyperTerminal and return the corresponding value.

2.3.1 Application main routine

The main application routine configures the peripherals and enables all the standard interrupts used by the application. When the initialization is complete, the main routine displays the application menu by calling the **App_Menu()** function in an infinite loop (see *Figure 4*). The main routine is exited only by a reset.

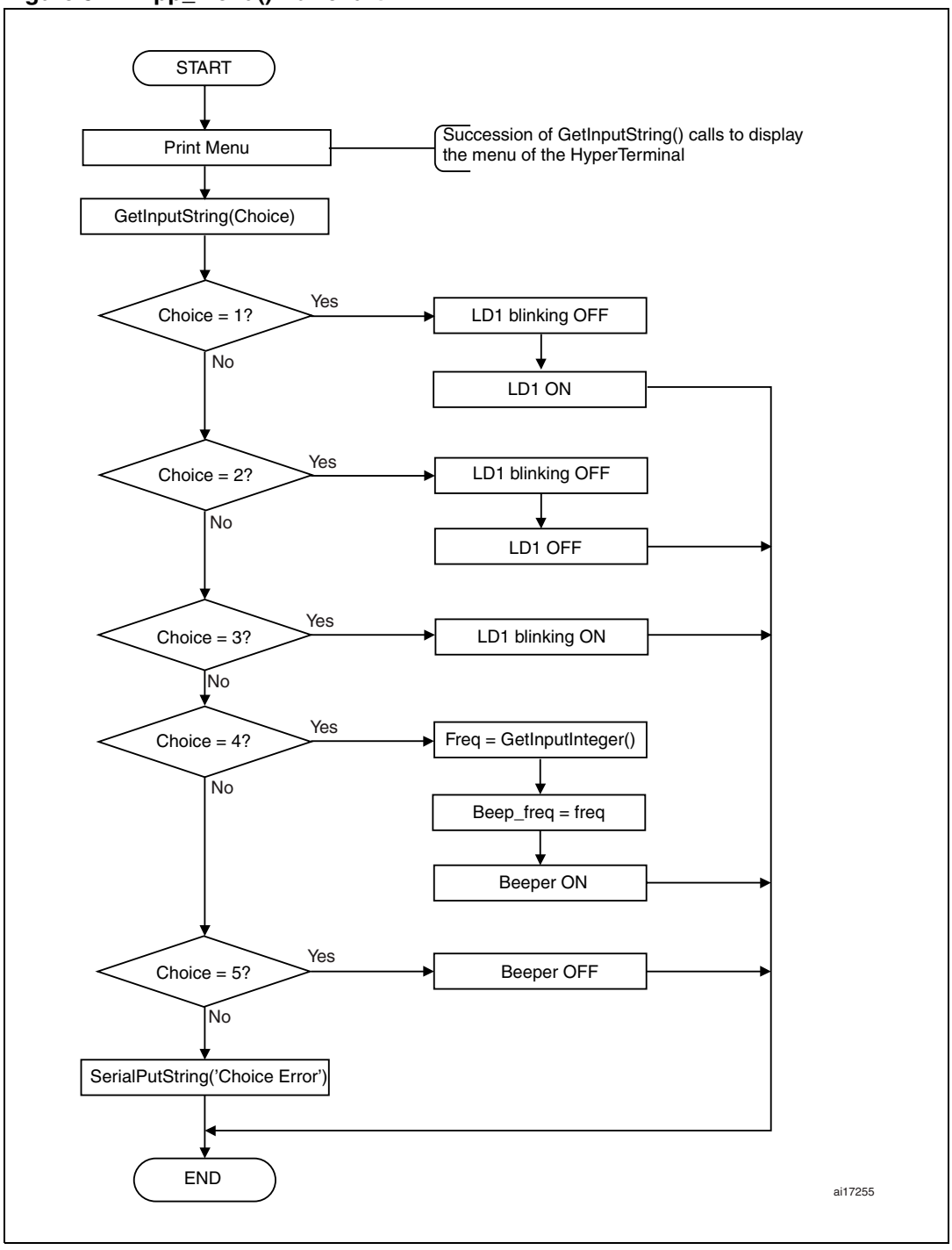
Figure 4. Main routine flowchart



2.3.2 App_menu () function

The App_menu() function sends and receives information from/to the Windows HyperTerminal using the STM8S UART interface. This function displays a menu interface on the HyperTerminal through which the GPIO, TIM2 and beeper can be configured. App_menu() calls GetInputString(), GetInputInteger() and SerialPutString() to send and received data through the RS232 interface.

Figure 5. App_menu() flowchart



2.3.3 GetInputString() function

The GetInputString() function receives and stores character strings send from the Windows HyperTerminal. This function calls the Get_key() function to read the ASCII codes received on UART2 (see *Section 2.3.4*), and stores the data in the Str string constant. Different actions may be performed according to the value of the ASCII code:

- If ascii_val = '\b'

  A backspace request has been sent by the HyperTerminal. The last character of the Str string is erased if Str is not empty.

- If ascii_val belongs to {0...1 or a...Z}

  The character is stored into the Str string.

- If ascii_val = '\r'

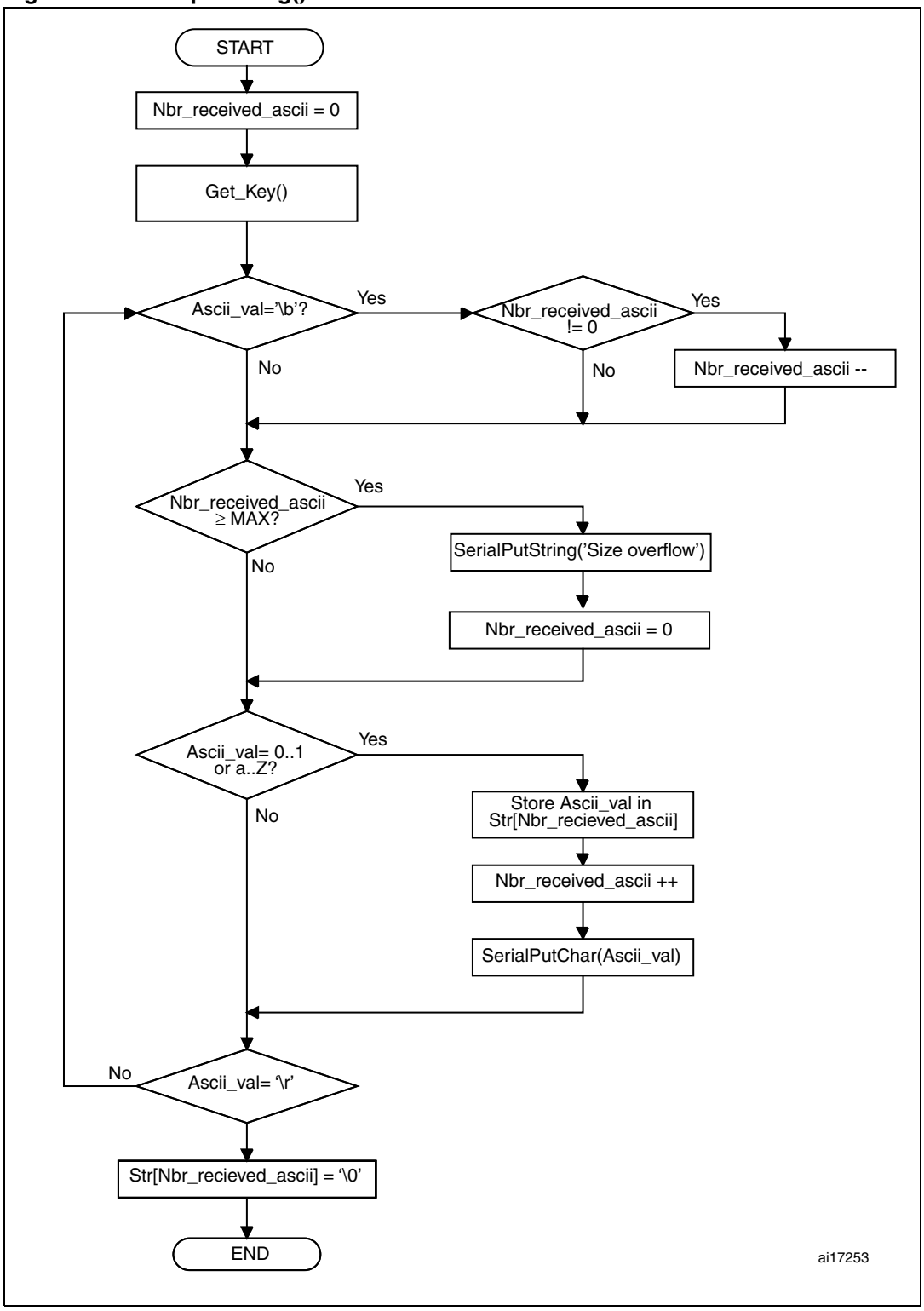
  The GetInputString() function stores the "end of string value", '\0', at the end of the Str string and stops.

- The maximum number of ASCII codes stored in Str (ascii_val) has been reached

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

For more information on ASCII codes refer to *Appendix A: Standard ASCII character codes*.

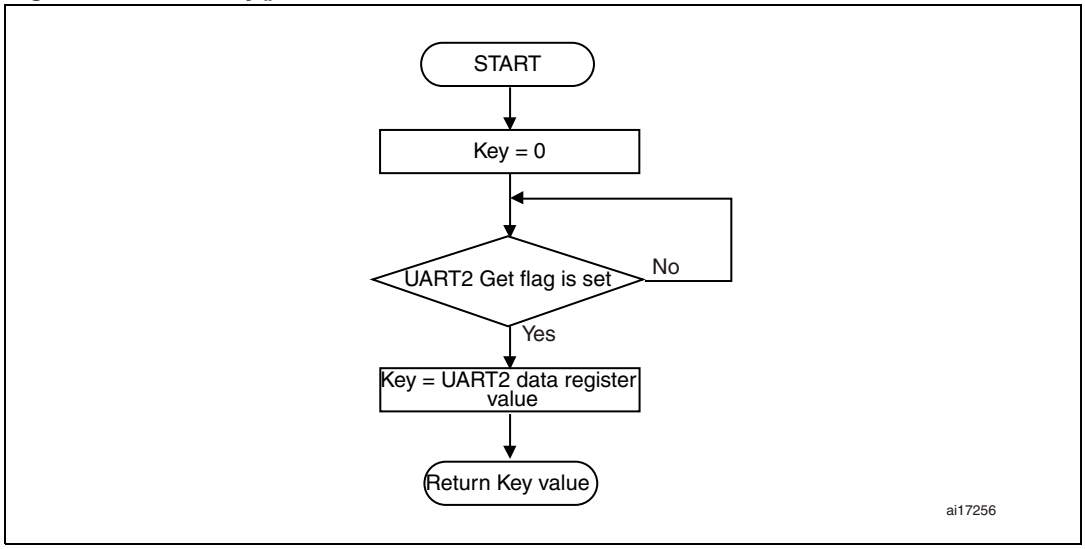
Figure 6. GetInputString() flowchart



2.3.4 Get_key() function

The Get_key() function is used to detect a key stroke on HyperTerminal by polling the UART Get flag. This function returns the received value.

Figure 7. Get_key() function flowchart



2.3.5 SerialPutChar() and SerialPutString() functions

The SerialPutChar() and SerialPutString() functions calls the firmware library UART2sendData8() function to send a char and wait for the UART datasend flag. SerialPutString() only performs a call to SerialPutChar() for all the characters of the string to be sent.

Figure 8. SerialPutChar() flowchart

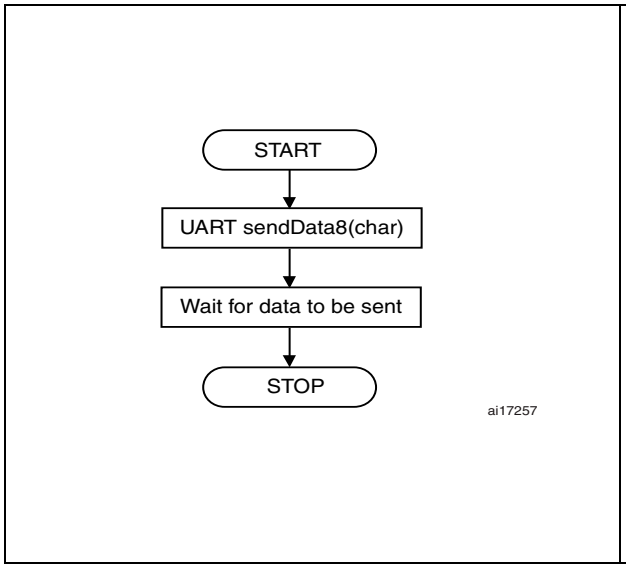
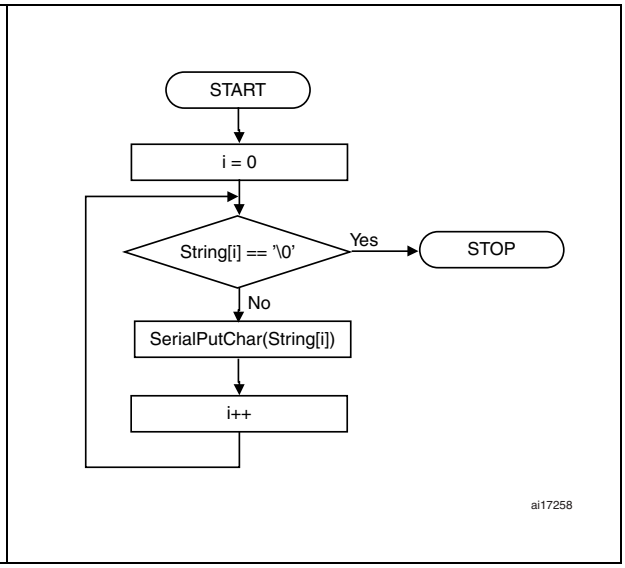


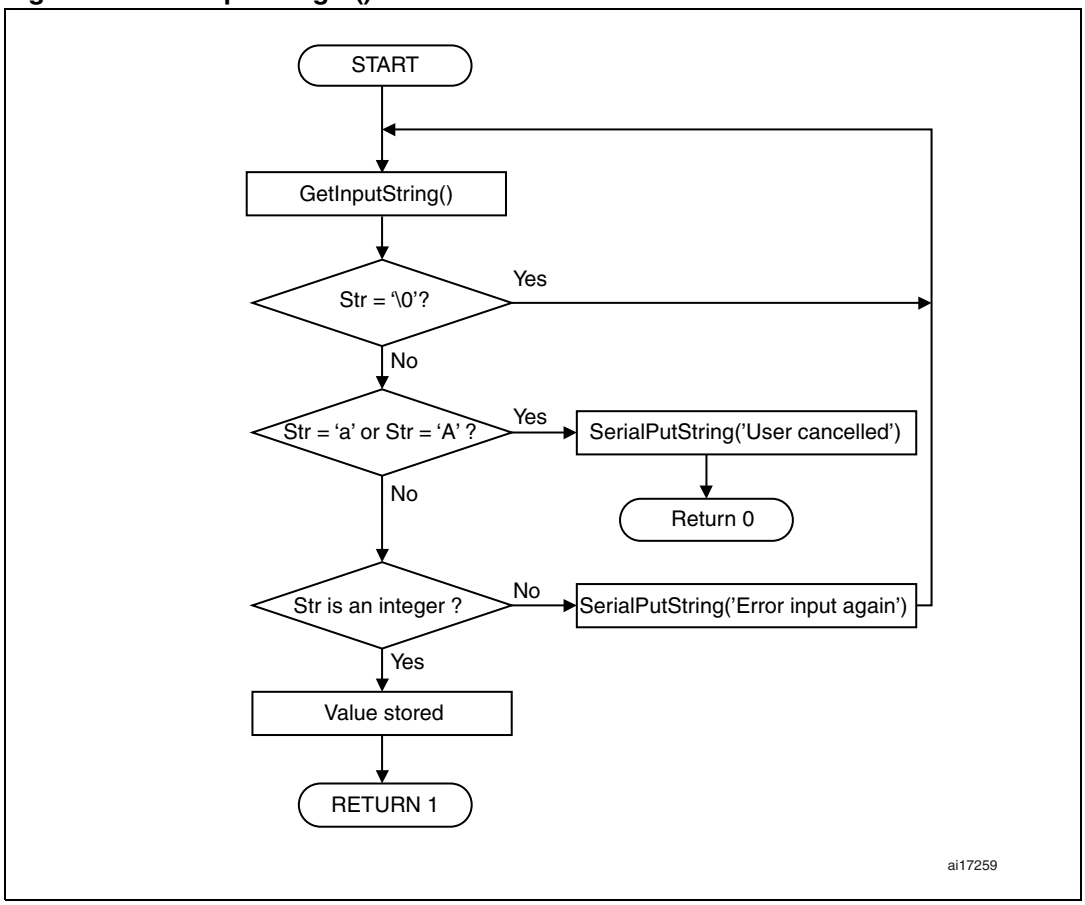
Figure 9. SerialPutString() flowchart



2.3.6 GetInputInteger() function

The GetInputInteger() function calls the GetInputString() function to get the transmitted string. If this string corresponds to ‘a’ or ‘A’, the operation is cancelled, otherwise the function checks if the string corresponds to an integer. If so, it stores the integer value in memory, otherwise it asks for an other input.

Figure 10. GetInputInteger() flowchart



Appendix A Standard ASCII character codes

Table 3. 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

Table 3. Standard ASCII character codes (continued)

Hex	Char	Hex	Char	Hex	Char	Hex	Char
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>

Appendix B HyperTerminal configuration

To configure the Windows HyperTerminal for your STM8S-DISCOVERY application, follow the steps below:

1. Launch Windows HyperTerminal, enter your application name and select an icon.

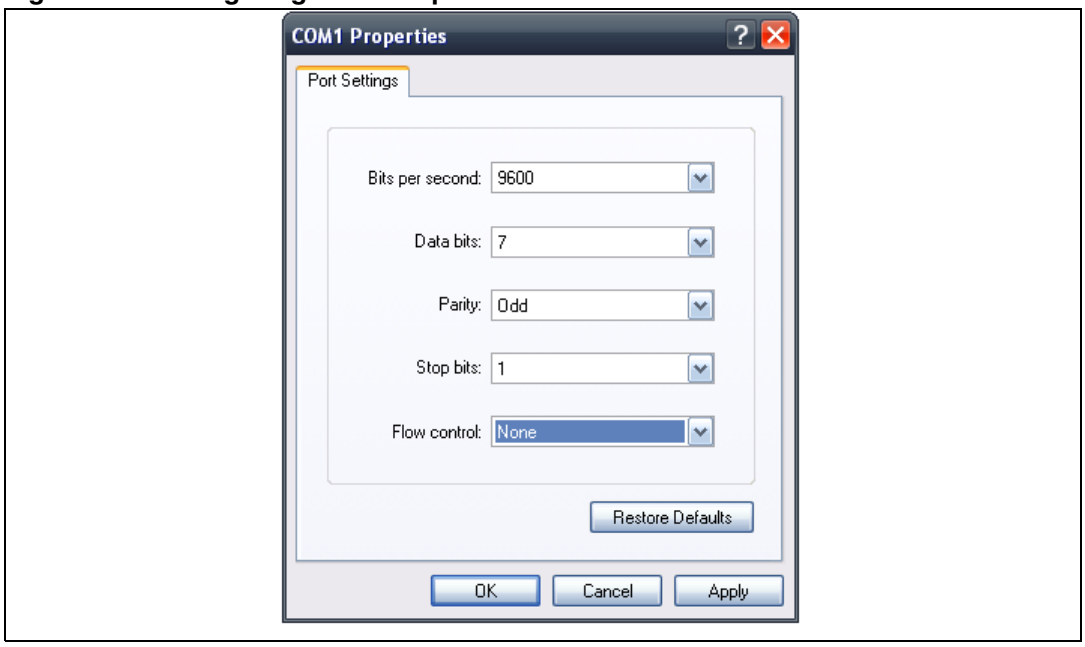
Figure 11. Connection window

2. Select the COM port

Figure 12. COM port selection

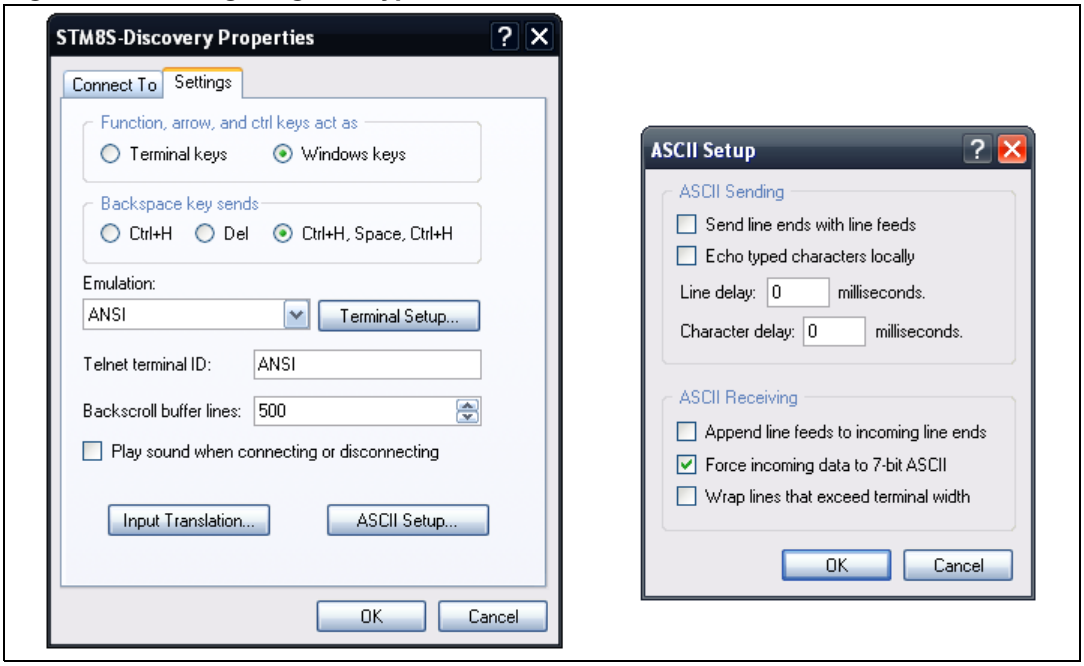
3. Configure the COM port in the same way as your UART. For the HyperTerminal application, the communications are configured as follows:

Figure 13. Configuring the COM port



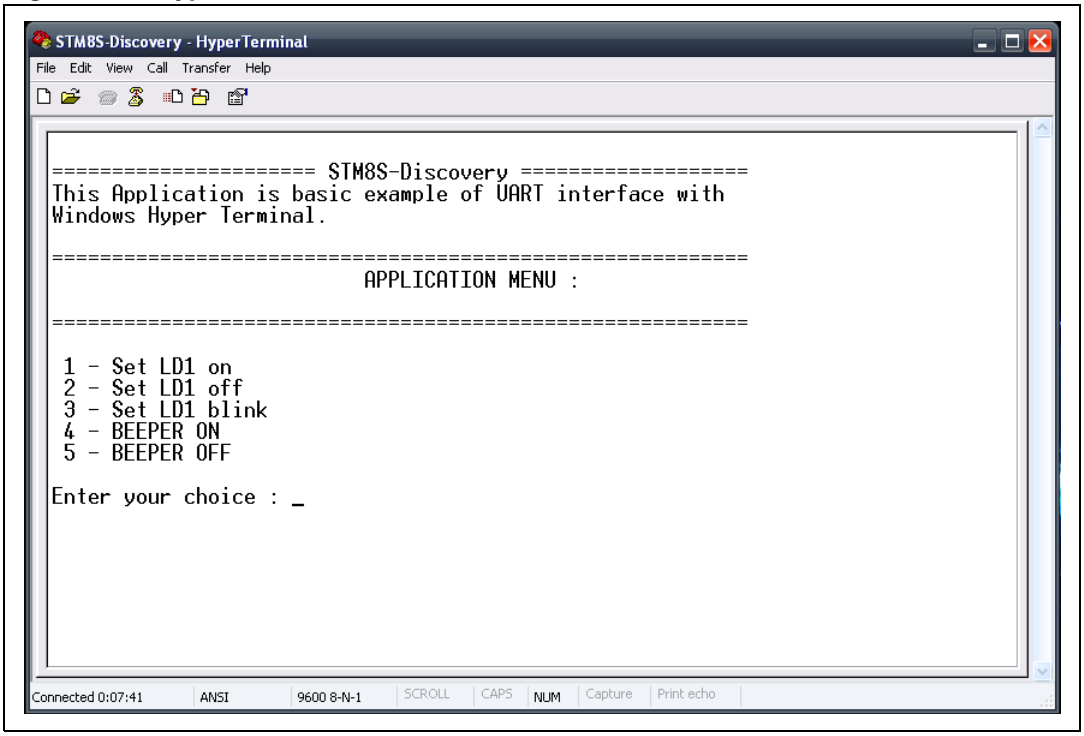
4. Configure the Windows HyperTerminal by selecting **File>Settings**:
   - Set **Backspace key** to Ctrl + H,space,Ctrl + H.
   - Set **Telnet terminal** ID to ANSI.
   - Unchek all **ASCII sending** options in the **ASCII Setup** window.
   - Check **Force incoming data to 7 bit ASCII**

Figure 14. Configuring the HyperTerminal



5. Connecting your STM8S-DISCOVERY board and launch the application using STVD. The HyperTerminal window is display. If it is not or if wrong characters are displayed restart the process starting from step 1.

Figure 15. HyperTerminal window



Revision history

Table 4. Document revision history

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

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com