



## Introduction

The bootloader is stored in the internal boot ROM memory (system memory) of STM32 devices. It is programmed by ST during production. Its main task is to download the application program to the internal Flash memory through one of the available serial peripherals (USART, CAN, USB, etc.). A communication protocol is defined for each serial interface, with a compatible command set and sequences.

The main features of the bootloader are the following:

- It uses an embedded serial interface to download the code with a predefined communication protocol
- It transfers and updates the Flash memory code, the data, and the vector table sections

This application note presents the general concept of the bootloader. It describes the supported peripherals and hardware requirements to be considered when using the bootloader of any STM32 device currently in production. However the specifications of the low-level communication protocol for each supported serial peripheral are documented in separate documents. For specifications of the USART protocol used in the bootloader please refer to AN3155. For the specification of CAN protocol used in the bootloader please refer to AN3154. For the specification of DFU (USB Device) protocol used in the bootloader please refer to AN3156.

# Contents

<b>1</b>	<b>Related documents</b>	<b>6</b>
<b>2</b>	<b>Glossary</b>	<b>7</b>
<b>3</b>	<b>General bootloader description</b>	<b>8</b>
3.1	Bootloader activation	8
3.2	Exiting System memory boot mode	8
3.3	Bootloader identification	8
<b>4</b>	<b>STM32F101xx, STM32F102xx, STM32F103xx, medium-density and high-density value line bootloader</b>	<b>10</b>
4.1	Bootloader configuration	10
4.2	Bootloader hardware requirements	11
4.3	Bootloader selection	11
4.4	Bootloader version	12
<b>5</b>	<b>STM32F105xx and STM32F107xx device bootloader</b>	<b>13</b>
5.1	Bootloader configuration	13
5.2	Bootloader hardware requirements	15
5.3	Bootloader selection	16
5.4	Bootloader version	18
<b>6</b>	<b>STM32F101xx and STM32F103xx XL-density device bootloader</b>	<b>19</b>
6.1	Dual bank boot feature	19
6.2	Bootloader configuration	21
6.3	Bootloader hardware requirements	22
6.4	Bootloader selection	23
6.5	Bootloader version	25
<b>7</b>	<b>STM32L15xx Medium-density Ultralow power device bootloader</b>	<b>26</b>
7.1	Bootloader configuration	26
7.2	Bootloader hardware requirements	27
7.3	Bootloader selection	27

---

7.4	Important considerations .....	29
7.5	Bootloader version .....	30
<b>8</b>	<b>STM32F205/215xx, and STM32F207/217xx bootloader .....</b>	<b>31</b>
8.1	Bootloader V2.x .....	31
8.1.1	Bootloader configuration .....	31
8.1.2	Bootloader hardware requirements .....	32
8.1.3	Bootloader selection .....	33
8.1.4	Important considerations .....	35
8.1.5	Bootloader V2.x versions .....	36
8.2	Bootloader V3.x .....	37
8.2.1	Bootloader configuration .....	37
8.2.2	Bootloader hardware requirements .....	39
8.2.3	Bootloader selection .....	40
8.2.4	Important considerations .....	43
8.2.5	Bootloader version V3.x .....	44
<b>9</b>	<b>Device-dependent bootloader parameters .....</b>	<b>45</b>
<b>10</b>	<b>Bootloader timing characteristics .....</b>	<b>46</b>
10.1	USART bootloader timing characteristics .....	46
10.2	USB bootloader timing characteristics .....	51
<b>11</b>	<b>Revision history .....</b>	<b>53</b>

## List of tables

Table 1.	Boot pin configuration . . . . .	8
Table 2.	Embedded bootloaders . . . . .	9
Table 3.	STM32F10xxx configuration in System memory boot mode . . . . .	10
Table 4.	STM32F10xxx bootloader versions . . . . .	12
Table 5.	STM32F105xx/107xx configuration in System memory boot mode . . . . .	13
Table 6.	STM32F105xx and STM32F107xx bootloader versions . . . . .	18
Table 7.	Boot pin and BFB2 bit configuration . . . . .	20
Table 8.	STM32F10xxx XL-density configuration in System memory boot mode . . . . .	21
Table 9.	XL-density bootloader versions . . . . .	25
Table 10.	STM32L15xxx configuration in System memory boot mode . . . . .	26
Table 11.	STM32L15xxx bootloader versions . . . . .	30
Table 12.	STM32F2xx configuration in System memory boot mode . . . . .	31
Table 13.	STM32F2xx Voltage Range configuration using bootloader V2.x . . . . .	36
Table 14.	STM32F2xx bootloader V2.x versions . . . . .	36
Table 15.	STM32F2xx configuration in System memory boot mode . . . . .	37
Table 16.	STM32F2xx Voltage Range configuration using bootloader V3.x . . . . .	44
Table 17.	STM32F2xx bootloader V3.x versions . . . . .	44
Table 18.	Bootloader device-dependant parameters . . . . .	45
Table 19.	USART bootloader timings for low/medium/high-density and value line devices . . . . .	47
Table 20.	USART bootloader timings for XL-density line devices . . . . .	48
Table 21.	USART bootloader timings for connectivity line devices (PA9 pin low) . . . . .	49
Table 22.	USART bootloader timings for connectivity line devices (PA9 high) . . . . .	49
Table 23.	USART bootloader timings for STM32L15xx medium-density ultralow power devices . . . . .	50
Table 24.	USART bootloader timings for STM32F205/215xx and STM32F207/217xx devices . . . . .	50
Table 25.	USB minimum timings for connectivity line devices . . . . .	52
Table 26.	USB minimum timings for STM32F205/215xx, and STM32F207/217xx devices . . . . .	52
Table 27.	Document revision history . . . . .	53

## List of figures

Figure 1.	Bootloader for STM32F10xxx with USART1 .....	11
Figure 2.	Bootloader selection for STM32F105xx and STM32F107xx devices .....	17
Figure 3.	Bootloader selection for STM32F10xxx XL-density devices .....	24
Figure 4.	Bootloader selection for STM32L15xxx devices .....	28
Figure 5.	Bootloader V2.x selection for STM32F2xx .....	34
Figure 6.	Bootloader V3.x selection for STM32F2xx .....	42
Figure 7.	USART bootloader timing waveforms .....	47
Figure 8.	USB bootloader timing waveforms .....	51

# 1 Related documents

All the documents mentioned below are available from [www.st.com](http://www.st.com):

- Datasheets
  - Low, medium and high-density STM32F101xx and STM32F103xx datasheets
  - Low, medium and high-density STM32F100xx and STM32F102xx datasheets
  - STM32F105xx/107xx connectivity line datasheet
  - XL-density STM32F101xx and STM32F103xx datasheets
  - STM32L151xx and STM32F152xx datasheet
  - STM32F205xx STM32F207xx and STM32F215xx STM32F217xx datasheets
- Reference manuals
  - STM32F101xx, STM32F102xx, STM32F103xx and STM32F105xx/107xx reference manual (RM0008)
  - Low, medium and high-density STM32F100xx value line reference manual (RM0041)
  - STM32L151xx and STM32L152xx advanced ARM-based 32-bit MCUs reference manual (RM0038)
  - STM32F205xx, STM32F207xx, STM32F215xx and STM32F217xx advanced ARM-based 32-bit MCUs reference manual (RM00033)
- Flash programming manuals
  - STM32F101xx, STM32F102xx, STM32F103xx and STM32F105xx/107xx Flash programming manual (PM0042)
  - Low, medium and high-density STM32F100xx value line Flash programming manual (PM0063)
  - XL-density STM32F101xx and STM32F103xx Flash programming manual (PM0068)
  - STM32L151xx and STM32L152xx Flash programming manual (PM0062)
  - STM32F205xx, STM32F207xx, STM32F215xx and STM32F217xx Flash programming manual (PM0059)

## 2 Glossary

**Low-density devices** are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

**Medium-density devices** are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

**High-density devices** are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

**Connectivity line devices** are STM32F105xx and STM32F107xx microcontrollers.

**Low-density value line devices** are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

**Medium-density value line devices** are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

**High-density value line devices** are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

**XL-density devices** are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 768 Kbytes and 1 Mbyte.

**Medium-density ultralow power devices** are STM32L15xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

**STM32F2xxx devices** are STM32F215xx, STM32F205xx, STM32F207xx and STM32F217xx microcontrollers with a Flash memory density ranging from 128 to 1024 Kbytes.

## 3 General bootloader description

### 3.1 Bootloader activation

The bootloader is automatically activated by configuring the BOOT0 and BOOT1 pins in the specific “System memory” configuration (see [Table 1](#)) and then by applying a reset.

Depending on the used pin configuration, the Flash memory, system memory or SRAM is selected as the boot space, as shown in [Table 1](#) below.

**Table 1. Boot pin configuration**

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
X	0	User Flash memory	User Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

[Table 1](#) shows that the STM32 microcontrollers enter the System memory boot mode if the BOOT pins are configured as follows:

- BOOT0 = 1
- BOOT1 = 0

The values on the BOOT pins are latched on the fourth rising edge of SYSCLK after a reset.

### 3.2 Exiting System memory boot mode

System memory boot mode must be exited in order to start execution of the application program. This can be done by applying a hardware reset. During reset, the BOOT pins (BOOT0 and BOOT1) must be set at the proper levels to select the desired boot mode (see [Table 1](#)). Following the reset, the CPU starts code execution from the boot memory located at the bottom of the memory address space starting from 0x0000 0000.

### 3.3 Bootloader identification

Depending on the STM32 device used, the bootloader may support one or more embedded serial peripherals used to download the code to the internal Flash memory. The bootloader identifier (ID) provides information about the supported serial peripherals.



For a given STM32 device, the bootloader is identified by means of the:

1. **Bootloader (protocol) version:** version of the serial peripheral (USART, CAN, USB, etc.) communication protocol used in the bootloader. This version can be retrieved using the bootloader Get Version command.
2. **Bootloader identifier (ID):** version of the STM32 device bootloader, coded on one byte in the **0xXY** format, where:
  - **X** specifies the embedded serial peripheral(s) used by the device bootloader:
    - X = 1: only one USART is used
    - X = 2: two USARTs are used
    - X = 3: two USARTs, one CAN and DFU are used
  - **Y** specifies the device bootloader version

Let us take the example of a bootloader ID equal to 0x10. This means that it is the first version of the device bootloader that uses only one USART.

The bootloader ID is programmed in the last two bytes of the device system memory and can be read by using the bootloader “Read memory” command or by direct access to the system memory via JTAG/SWD.

The table below provides identification information about the bootloader embedded in STM32 devices.

**Table 2. Embedded bootloaders**

Device	Supported serial peripherals	Bootloader ID		Bootloader (protocol) version
		ID	Memory location	
Low-density	USART1	NA	NA	USART (V2.2)
Medium-density	USART1	NA	NA	USART (V2.2)
High-density	USART1	NA	NA	USART (V2.2)
Connectivity line	USART1 / USART2 (remapped) / CAN2 (remapped) / DFU (USB Device)	NA	NA	USART (V2.2 <sup>(1)</sup> ) CAN (V2.0) DFU (V2.0)
Medium-density value line	USART1	V1.0	0x1FFFF7D6	USART (V2.2)
High-density value line	USART1	V1.0	0x1FFFF7D6	USART (V2.2)
XL-density	USART1/USART2 (remapped)	V2.1	0x1FFFF7D6	USART (V3.0)
Medium-density ultralow power line	USART1/USART2	V2.0	0x1FF00FFE	USART (V3.0)
STM32F2xxx devices	USART1/USART3	V2.0	0x1FFF77DE	USART (V3.0)
	USART1/USART3/CAN2/DFU (USB Device FS)	V3.2	0x1FFF77DE	USART (V3.0)/ CAN (V2.0)/ DFU (V2.1)

1. For connectivity line devices, the USART bootloader returns V2.0 instead of V2.2 for the protocol version. For more details please refer to the "STM32F105xx and STM32F107xx revision Z" errata sheet available from [www.st.com](http://www.st.com).

## 4 STM32F101xx, STM32F102xx, STM32F103xx, medium-density and high-density value line bootloader

Throughout this section **STM32F10xxx** will be used to refer to low-density, medium-density, high-density STM32F101xx and STM32F103xx devices, to low- and medium-density STM32F102xx devices and to medium and high-density value line devices.

### 4.1 Bootloader configuration

The bootloader embedded in STM32F10xxx devices supports only one interface: the USART1.

The following table shows the required STM32F10xxx hardware resources used by the bootloader in System memory boot mode.

**Table 3. STM32F10xxx configuration in System memory boot mode**

Feature/Peripheral	State	Comment
Clock source	HSI enabled	The system clock is equal to 24 MHz using the PLL
USART1_RX pin	Input	PA10 pin: USART1 receives
USART1_TX pin	Output	PA9 pin: USART1 transmits
SysTick timer	Enabled	Used to automatically detect the serial baud rate from the host.
USART1	Enabled	Once initialized the USART1 configuration is: 8-bits, even parity and 1 Stop bit
RAM	-	512 bytes starting from address 0x2000 0000 are used by the bootloader firmware
System memory	-	2 Kbytes starting from address 0x1FFF F000, contain the bootloader firmware
IWDG	-	The independent watchdog (IWDG) prescaler is configured to its maximum value and is periodically refreshed to prevent watchdog reset (in case the hardware IWDG option was previously enabled by the user)

The system clock is derived from the embedded internal high-speed RC, no external quartz is required for the bootloader code.

After downloading the application binary, if you choose to execute the Go command, the peripheral registers used by the bootloader (shown in the above table) are not initialized to their default reset values before jumping to the user application. They should be reconfigured in the user application if they are used. So, if the IWDG is being used in the application, the IWDG prescaler value has to be adapted to meet the requirements of the application (since the prescaler was set to its maximum value by the bootloader).

## 4.2 Bootloader hardware requirements

The hardware required to put the STM32 into System memory boot mode consists of any circuitry, switch or jumper, capable of holding the BOOT0 pin high and the BOOT1 pin low during reset.

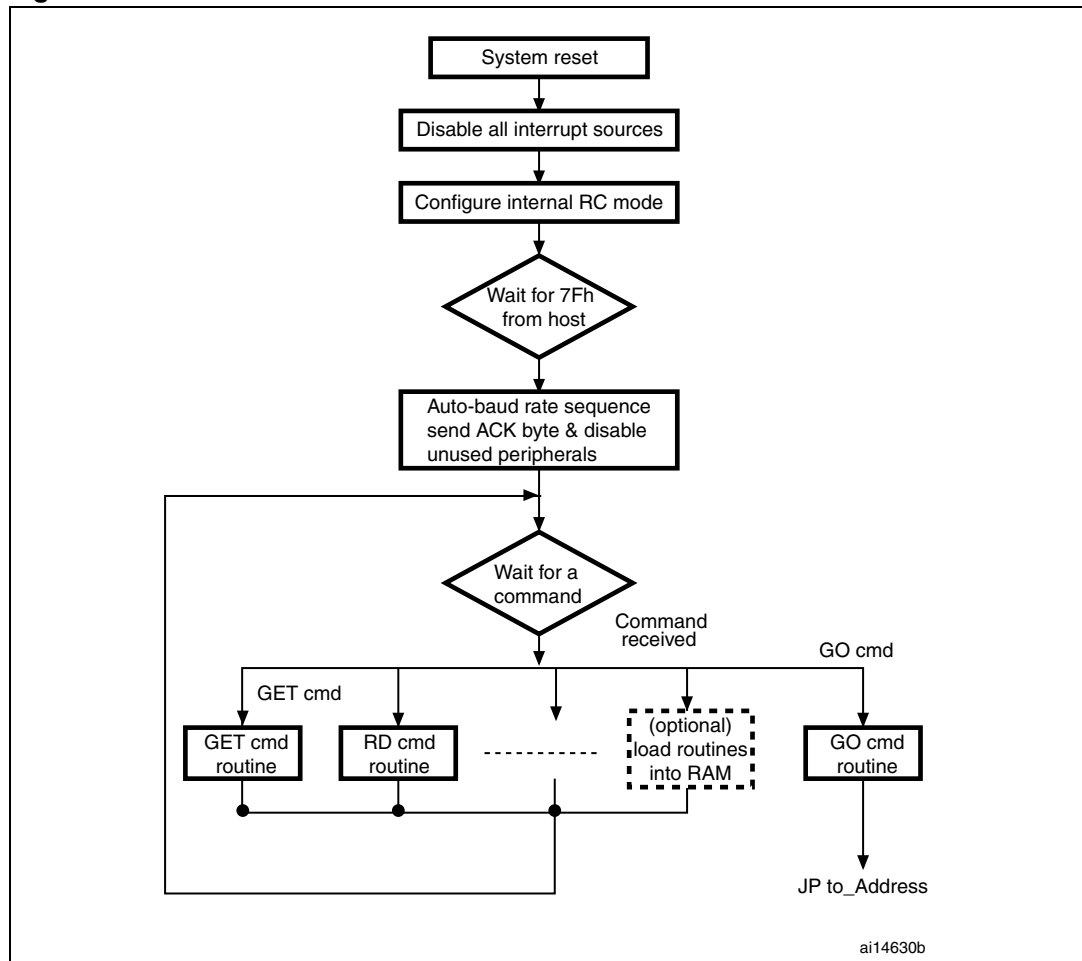
To connect to the STM32 during System memory boot mode, an RS232 serial interface (example, ST3232 RS232 transceiver) has to be directly linked to the USART1\_RX (PA10) and USART1\_TX (PA9) pins.

*Note:* **USART1\_CK, USART1\_CTS and USART1\_RTS pins are not used, therefore user can use these pins for other peripherals or GPIOs.**

*For more details about hardware recommendations, refer to application note AN2586: "STM32 hardware development: getting started", available from the STMicroelectronics website: [www.st.com](http://www.st.com).*

## 4.3 Bootloader selection

**Figure 1. Bootloader for STM32F10xxx with USART1**



Once System memory boot mode is entered and the microcontroller has been configured as described above, the bootloader code begins to scan the USART1\_RX line pin, waiting to receive the 0x7F data frame: one start bit, 0x7F data bits, even parity bit and one stop bit.

The duration of this data frame is measured using the SysTick timer. The count value of the timer is then used to calculate the corresponding baud rate factor with respect to the current system clock.

Next, the code initializes the serial interface accordingly. Using this calculated baud rate, an acknowledge byte (0x79) is returned to the host, which signals that the STM32F10xxx is ready to receive user commands.

## 4.4 Bootloader version

[Table 4](#) lists the bootloader versions of the STM32F10xxx devices.

**Table 4. STM32F10xxx bootloader versions**

Bootloader version number	Description
V2.0	Initial bootloader version.
V2.1	<ul style="list-style-type: none"> <li>– Update Go Command to initialize the main stack pointer</li> <li>– Update Go command to return NACK when jump address is in the Option byte area or System memory area</li> <li>– Update Get ID command to return the device ID on two bytes</li> <li>– Update the bootloader version to V2.1</li> </ul>
V2.2	<ul style="list-style-type: none"> <li>– Update Read Memory, Write Memory and Go commands to deny access with a NACK response to the first 0x200 bytes of RAM memory used by the bootloader</li> <li>– Update Readout Unprotect command to initialize the whole RAM content to 0x0 before ROP disable operation</li> </ul>

## 5 STM32F105xx and STM32F107xx device bootloader

### 5.1 Bootloader configuration

The bootloader embedded in the STM32F105xx and STM32F107xx devices supports four serial peripherals: USART1, USART2, CAN2, and DFU (USB). This means that four serial peripherals are supported: USART1, USART2, CAN2 and DFU (USB).

The following table shows the hardware resources required by STM32F105xx and STM32F107xx devices used by the bootloader in System memory boot mode.

**Table 5. STM32F105xx/107xx configuration in System memory boot mode**

Bootloader	Feature/Peripheral	State	Comment
Common to all bootloaders	RCC	HSI enabled	The system clock frequency is 24 MHz using the PLL. This is used only for USART1 and USART2 bootloaders and during CAN2, USB detection for CAN and DFU bootloaders (Once CAN or DFU bootloader is selected, the clock source will be derived from external crystal).
		HSE enabled	The external clock is mandatory only for DFU and CAN bootloaders and it must provide one of the following frequencies: 8 MHz, 14.7456 MHz or 25 MHz. For CAN Bootloader, the PLL is used only to generate 48 MHz when 14.7456 MHz is used as HSE. For DFU Bootloader, the PLL is used to generate a 48 MHz system clock from all supported external clock frequencies.
		-	The clock security system (CSS) interrupt is enabled for the CAN and DFU bootloaders. Any failure (or removal) of the external clock will generate system reset.
	IWDG	-	The independent watchdog (IWDG) prescaler is configured to its maximum value and is periodically refreshed to prevent watchdog reset (in case the hardware IWDG option was previously enabled by the user).
	System memory	-	18 Kbytes starting from address 0x1FFF B000, contain the bootloader firmware
	RAM	-	4 Kbytes starting from address 0x2000 0000 are used by the bootloader firmware.
USART1 bootloader	USART1	Enabled	Once initialized the USART1 configuration is: 8-bits, even parity and 1 Stop bit
	USART1_RX pin	Input	PA10 pin: USART1 receive
	USART1_TX pin	Output	PA9 pin: USART1 transmit
	USART2_RX (PD6), CAN2_RX (PB5), OTG_FS_DM (PA11) and OTG_FS_DP (PA12) pins must be kept at a high or low level during the detection phase.		

Table 5. STM32F105xx/107xx configuration in System memory boot mode (continued)

Bootloader	Feature/Peripheral	State	Comment
USART1 and USART2 bootloaders	SysTick timer	Enabled	Used to automatically detect the serial baud rate from the host for USARTx bootloader.
USART2 bootloader	USART2	Enabled	Once initialized the USART2 configuration is: 8-bits, even parity and 1 Stop bit. The USART2 uses its remapped pins.
	USART2_RX pin	Input	PD6 pin: USART2 receive (remapped pin)
	USART2_TX pin	Output	PD5 pin: USART2 transmit (remapped pin)
	USART1_RX (PA10), CAN2_RX (PB5), OTG_FS_DM (PA11) and OTG_FS_DP (PA12) pins must be kept at a high or low level during the detection phase.		
CAN2 bootloader	CAN2	Enabled	Once initialized the CAN2 configuration is: Baudrate 125 kbps, 11-bit identifier. <b>Note:</b> CAN1 is clocked during the CAN bootloader execution because in STM32F105xx and STM32F107xx devices, CAN1 manages the communication between CAN2 and SRAM.
	CAN2_RX pin	Input	PB5 pin: CAN2 receives (remapped pin)
	CAN2_TX pin	Output	PB6 pin: CAN2 transmits (remapped pin)
	USART1_RX (PA10), USART2_RX (PD6), OTG_FS_DM (PA11) and OTG_FS_DP (PA12) pins must be kept at a high or low level during the detection phase.		
DFU bootloader	USB OTG FS	Enabled	USB OTG FS configured in Forced Device mode
	OTG_FS_VBUS pin	Input or alternate function, automatically controlled by the USB OTG FS controller	PA9: Power supply voltage line
	OTG_FS_DM pin		PA11: USB Send-Receive data line
	OTG_FS_DP pin		PA12: USB Send-Receive data line
	Interrupts	Enabled	USB_OTG_FS interrupt vector is enabled and used for USB DFU communication.
	USART1_RX (PA10), USART2_RX (PD6) and CAN2_RX (PB5) pins must be kept at a high or low level during the detection phase.		

The system clock is derived from the embedded internal high-speed RC for USARTx bootloader. This internal clock is used also for DFU and CAN bootloaders but only for the selection phase. An external clock (8 MHz, 14.7456 MHz or 25 MHz.) is required for DFU and CAN bootloader execution after the selection phase.

After downloading the application binary, if you choose to execute the Go command, all peripheral registers used by the bootloader (shown in the above table) will be initialized to their default reset values before jumping to the user application.  
If the user application uses the IWDG, the IWDG prescaler value has to be adapted to meet the requirements of the application (since the prescaler was set to its maximum value by the bootloader).

## 5.2 Bootloader hardware requirements

The hardware required to put the STM32F105xx and STM32F107xx into System memory boot mode consists of any circuitry, switch or jumper, capable of holding the BOOT0 pin high and the BOOT1 pin low during reset.

To connect to the STM32F105xx and STM32F107xx during System memory boot mode, the following conditions have to be verified:

- The RX pins of the unused peripherals in this bootloader have to be kept at a known (low or high) level, and should not be left floating during the detection phase as described below:
  - If USART1 is used to connect to the bootloader: the USART2\_RX (PD6), CAN2\_RX (PB5), OTG\_FS\_DM (PA11) and OTG\_FS\_DP (PA12) pins have to be kept at a high or low level and must not be left floating during the detection phase.
  - If USART2 is used to connect to the bootloader: the USART1\_RX (PA10), CAN2\_RX (PB5), OTG\_FS\_DM (PA11) and OTG\_FS\_DP (PA12) pins have to be kept at a high or low level and must not be left floating during the detection phase.
  - If CAN2 is used to connect to the bootloader: the USART1\_RX (PA10), USART2\_RX (PD6), OTG\_FS\_DM (PA11) and OTG\_FS\_DP (PA12) pins have to be kept at a high or low level and must not be left floating during the detection phase.
  - If DFU is used to connect to the bootloader: the USART1\_RX (PA10), USART2\_RX (PD6) and CAN2\_RX (PB5) pins have to be kept at a high or low level and must not be left floating during the detection phase.
- Connect to the peripheral to be used through:
  - an RS232 serial interface (example, ST3232 RS232 transceiver) has to be directly connected to the USART1\_RX (PA10) and USART1\_TX (PA9) pins when USART1 is used, or to the USART2\_RX (PD6) and USART2\_TX (PD5) pins when USART2 is used
  - a CAN interface (CAN transceiver) has to be directly connected to the CAN2\_RX (PB5) and CAN2\_TX (PB6) pins
  - a certified USB cable has to be connected to the microcontroller (optionally an ESD protection circuitry can be used)

The USART1\_CK, USART1\_CTS and USART1\_RTS pins are not used, therefore the application can use these pins for other peripherals or GPIOs. The same note is applicable for USART2.

Once the USB Device is enabled, all its related pins are dedicated to USB communication only, and cannot be used for other application purposes.

The user can control the BOOT0 and Reset pins from a PC serial applet using the RS232 serial interface which controls BOOT0 through the CTS line and Reset through the DCD line. The user must use a full null modem cable. The necessary hardware to implement for this control exists in the STM3210C-EVAL board. For more details about this, refer to document: “STM3210C-EVAL board user manual”, available from the STMicroelectronics website: [www.st.com](http://www.st.com).

## 5.3 Bootloader selection

The STM32F105xx and STM32F107xx embedded bootloader supports four peripherals interfaces: USART1, USART2, CAN2 and DFU (USB). Any one of these peripheral interfaces can be used to communicate with the bootloader and download the application code to the internal Flash.

The embedded bootloader firmware is able to auto-detect the peripheral interface to be used. In an infinite loop, it detects any communication on the supported bootloader interfaces.

**Note:** *The RX pins of the peripherals not used in this bootloader must be kept at a known (low or high) level and should not be left floating during the detection phase as described below. Refer to [Section 5.2: Bootloader hardware requirements](#) for more information.*

To use the USART bootloader on USART1 or USART2, connect the serial cable to the desired interface. Once the bootloader detects the data byte 0x7F on this interface, the bootloader firmware executes the auto-baud rate sequence and then enters a loop, waiting for any USART bootloader command.

To use the CAN2 interface, connect the CAN cable to CAN2. Once the bootloader detects a frame on the CAN2\_RX pin (PB5), the bootloader firmware enters a CAN loop and starts to check the external clock frequency value, if the HSE is 8 MHz, 14.7456 MHz or 25 MHz CAN bootloader firmware enters an infinite loop and waits until it receives a message, otherwise a system reset is generated.

If a USB cable is plugged into the microcontroller's USB interface at any time during the bootloader firmware selection sequence, the bootloader then enters the DFU bootloader loop waiting for any DFU bootloader command.

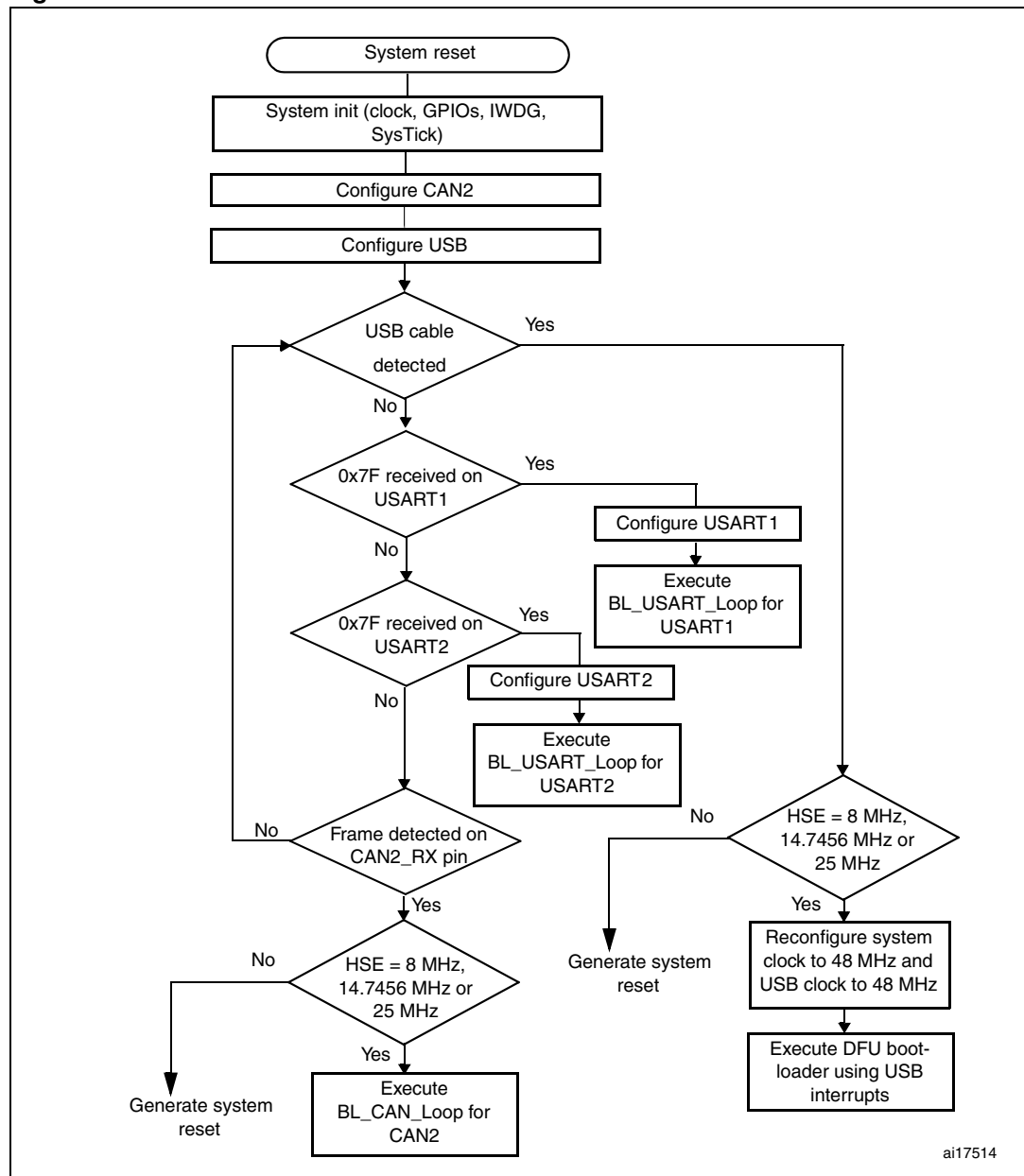
To use the USART or the CAN bootloader, it is mandatory that no USB cable is connected to the USB peripheral during the selection phase. Once the USART or CAN bootloader is selected, the user can plug a USB cable without impacting the selected bootloader execution except commands which generate a system reset.

Once one interface is selected for the bootloader, all other interfaces are disabled.

The figure below shows the bootloader detection mechanism. More details are provided in the sections corresponding to each peripheral bootloader.



Figure 2. Bootloader selection for STM32F105xx and STM32F107xx devices



## 5.4 Bootloader version

The table below lists the bootloader versions and the changes between versions V1.0 and V2.0. of the STM32F105xx and STM32F107xx devices

**Table 6. STM32F105xx and STM32F107xx bootloader versions**

Bootloader version number	Description
V1.0	Initial bootloader version.
V2.0	<ul style="list-style-type: none"><li>– Bootloader detection mechanism updated to fix the issue when GPIOs of unused peripherals in this bootloader are connected to low level or left floating during the detection phase. For more details please refer to limitation 2.12 “Boot loader unavailability on STM32F105xx and STM32F107xx devices with a date code below 937” as described in Revision 2 of the “STM32F105xx and STM32F107xx revision Z” errata sheet available from <a href="http://www.st.com">www.st.com</a></li><li>– Vector table set to 0x1FFF B000 instead of 0x0000 0000</li><li>– Go command updated (for all bootloaders): USART1, USART2, CAN2, GPIOA, GPIOB, GPIOD and SysTick peripheral registers are set to their default reset values</li><li>– DFU bootloader: USB pending interrupt cleared before executing the Leave DFU command</li><li>– DFU subprotocol version changed from V1.0 to V1.2</li><li>– Bootloader version updated to V2.0</li></ul>

## 6 STM32F101xx and STM32F103xx XL-density device bootloader

Throughout this section **STM32F10xxx XL-density** is used to refer to XL-density STM32F101xx and STM32F103xx devices.

### 6.1 Dual bank boot feature

For STM32F101xx and STM32F103xx XL-density devices (these devices have two Flash memory banks: Bank 1 and Bank 2), an additional boot mechanism is available which allows booting from Bank 2 or Bank 1 (depending on the BFB2 bit status (bit 19 in the user option bytes @ 0x1FFFF800)).

1. When the BFB2 bit is reset, and the boot pins are configured to boot from the Flash memory (BOOT0 = 0 and BOOT1 = x) then, after reset, the device boots from the System memory and executes the embedded bootloader code which implements the dual bank Boot mode:
  - a) First, the code checks Bank 2. If it contains a valid code (see [Note: 1](#) below), it jumps to application located in Bank 2 and leaves the Bootloader.
  - b) If the Bank 2 code is not valid, it checks Bank 1 code. If it is valid (see “*note*” below), it jumps to the application located in Bank 1.
  - c) If both Bank 2 and Bank 1 do not contain valid code (see “*note*” below), the normal Bootloader operations are executed as described in the following sections (no jump to Flash banks is executed). Refer to [Figure 3: Bootloader selection for STM32F10xxx XL-density devices](#) for more details.
2. When the bit BFB2 is set (default state), the dual bank boot mechanism is not performed.

- Note:*
- 1 The code is considered as valid when the first data (at the bank start address, which should be the stack pointer) points to a valid address into the internal SRAM memory (stack top address). If the first address points to any other location (out of the internal SRAM) the code is considered not valid.
  - 2 A dual bank Boot mode example (FLASH\Dual\_Boot) is provided within the STM32F10x Standard Peripheral Library available on [www.st.com](http://www.st.com).

For the STM32F101xx and STM32F103xx XL-density devices, the Flash memory, system memory or SRAM is selected as the boot space, as shown in [Table 7](#) below.

**Table 7. Boot pin and BFB2 bit configuration**

BFB2 bit	Boot mode selection pins		Boot mode	Aliasing
	BOOT1	BOOT0		
1	X	0	User Flash memory	User Flash memory is selected as the boot space
	0	1	System memory	System memory is selected as the boot space
	1	1	Embedded SRAM	Embedded SRAM is selected as the boot space
0	X	0	System memory	System memory is selected as the boot space then dual bank mechanism is executed
	0	1	System memory	System memory is selected as the boot space then dual bank mechanism is executed
	1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

[Table 7](#) shows that the XL-density devices enter the System memory boot mode in two cases:

1. If the BOOT pins are configured as follows: BOOT0 = 1 and BOOT1 = 0
2. Or if:
  - a) the BFB2 bit is reset and
  - b) boot pins are configured as follows: BOOT0 = 0 and BOOT1 = x

**Note:** When conditions a, b, and c below are fulfilled, it is equivalent to configuring boot pins for system memory boot (BOOT0 = 1 and BOOT1 = 0). In this case normal Bootloader operations are executed.

- a) BFB2 bit is reset
- b) Both banks don't contain valid code
- c) Boot pins configured as follows: BOOT0 = 0 and BOOT1 = x

*When the BFB2 bit is cleared, and Bank 2 and/or Bank 1 contain valid user application code, the Dual Bank Boot is always performed (bootloader always jumps to the user code and never continues normal operations).*

*Consequently, if you have cleared the BFB2 bit (to boot from Bank 2) then, to be able to execute the Bootloader code, you have to:*

*- either, set the BFB2 bit to 1*

*- or, program the content of address 0x0808 0000 (base address of Bank2) and 0x0800 0000 (base address of Bank1) to 0x0*

## 6.2 Bootloader configuration

The bootloader embedded in STM32F10xxx XL-density supports two serial interfaces: USART1 and USART2.

The following table shows the required hardware resources of STM32F10xxx XL-density devices used by the bootloader in System memory boot mode.

**Table 8. STM32F10xxx XL-density configuration in System memory boot mode**

Feature/peripheral	State	Comment
Clock source	HSI enabled	The system clock is equal to 24 MHz using the PLL
USART1_RX pin	Input	PA10 pin: USART1 receives
USART1_TX pin	Output	PA9 pin: USART1 transmits
USART2_RX pin	Input	PD6 pin: USART2 receives (remapped pins)
USART2_TX pin	Output	PD5 pin: USART2 transmits (remapped pins)
SysTick timer	Enabled	Used to automatically detect the serial baud rate from the host
USART1	Enabled	Once initialized the USART1/USART2 configuration is: 8-bits, even parity and 1 Stop bit.
USART2	Enabled	
RAM	-	2 Kbytes starting from address 0x2000 0000 are used by the bootloader firmware
System memory	-	6 Kbytes starting from address 0x1FFF E000, contain the bootloader firmware
IWDG	-	The independent watchdog (IWDG) prescaler is configured to its maximum value and is periodically refreshed to prevent watchdog reset (in case the hardware IWDG option was previously enabled by the user)

The system clock is derived from the embedded internal high-speed RC, no external quartz is required for the bootloader code.

After downloading the application binary, if you choose to execute the Go command, all peripheral registers used by the bootloader (shown in [Table 8](#)) are initialized to their default reset values before jumping to the user application.

If the user application uses the IWDG, the IWDG prescaler value has to be adapted to meet the requirements of the application (since the prescaler was set to its maximum value by the bootloader).

## 6.3 Bootloader hardware requirements

The hardware required to put the STM32F10xx XL-density devices into System memory boot mode consists of any circuitry, switch or jumper, capable of holding the BOOT0 pin high and the BOOT1 pin low during reset.

*Note:* As explained in [Section 6.1: Dual bank boot feature](#), the System memory boot mode can also be executed by software when the BFB2 bit is reset, both banks start addresses are erased, and boot pins are configured to boot from Flash memory.

To connect to the STM32F10xx XL-density devices during System memory boot mode, the following conditions have to be verified:

- The RX pin of the peripherals unused in this bootloader have to be kept at a known (low or high) level, and should not be left floating during the detection phase as described below:
  - If the USART1 is used to connect to the bootloader: the USART2\_RX (PD6) pin has to be kept at a high or low level and must not be left floating during the detection phase.
  - If the USART2 is used to connect to the bootloader: the USART1\_RX (PA10) pin has to be kept at a high or low level and must not be left floating during the detection phase.
- When the BFB2 bit is cleared, and Bank 2 and/or Bank 1 contain a valid user application code, the Dual Bank Boot is always performed (bootloader always jumps to the user code and never continues normal operations). Consequently, if you have cleared the BFB2 bit (to boot from Bank 2), then to be able to execute the Bootloader code, you have to:
  - either, set the BFB2 bit to 1
  - or, program the content of address 0x0808 0000 (base address of Bank2) and 0x0800 0000 (base address of Bank1) to 0x0
- Connect to the peripheral to be used through:
  - an RS232 serial interface (example, ST3232 RS232 transceiver) has to be directly connected to the USART1\_RX (PA10) and USART1\_TX (PA9) pins when USART1 is used, or to the USART2\_RX (PD6) and USART2\_TX (PD5) pins when USART2 is used

The USART1\_CK, USART1\_CTS and USART1\_RTS pins are not used, therefore the application can use these pins for other peripherals or GPIOs. This is also applicable for USART2.

## 6.4 Bootloader selection

The STM32F10xx XL-density embedded Bootloader supports two peripheral interfaces: USART1 and USART2. Any one of these peripheral interfaces can be used to communicate with the bootloader and download the application code to the internal Flash.

The embedded Bootloader firmware is able to auto-detect the peripheral interface to be used. In an infinite loop, it detects any communication on the supported bootloader interfaces.

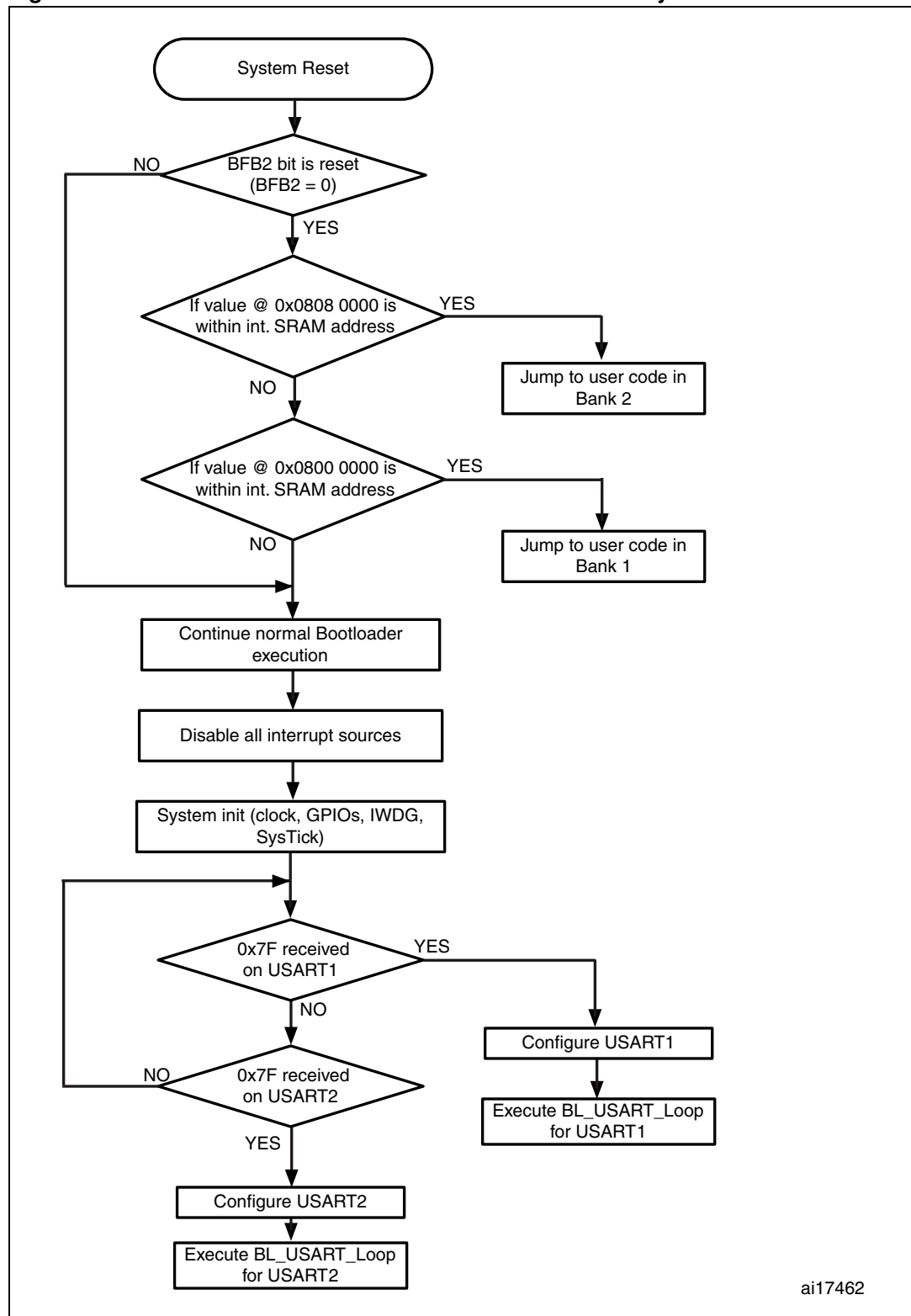
**Note:** *The RX pins of the peripherals not used in this bootloader must be kept at a known (low or high) level and should not be left floating during the detection phase as described below. Refer to [Section 6.3: Bootloader hardware requirements](#) for more information.*

To use the USART bootloader on USART1 or USART2, connect the serial cable to the desired interface. Once the bootloader detects the data byte 0x7F on this interface, the bootloader firmware executes the auto-baudrate sequence and then enters a loop, waiting for any USART bootloader command.

Once one interface is selected for the bootloader, the other interface is disabled.

[Figure 3](#) shows the bootloader detection mechanism. More details are provided in the sections corresponding to each peripheral bootloader.

Figure 3. Bootloader selection for STM32F10xxx XL-density devices





## 6.5 Bootloader version

[Table 9](#) lists the bootloader versions for the STM32F101xx and STM32F103xx XL-density devices.

**Table 9. XL-density bootloader versions**

Bootloader version number	Description
V2.1	Initial bootloader version

## 7 STM32L15xx Medium-density Ultralow power device bootloader

Through all this section **STM32L15xxx** will be used as reference to Medium-density STM32L151xx and STM32L152xx ultralow power devices.

### 7.1 Bootloader configuration

The bootloader embedded in STM32L15xxx devices supports two serial interfaces: USART1 and USART2 peripherals.

The following table shows the required hardware resources of STM32L15xx devices used by the bootloader in System memory boot mode.

**Table 10. STM32L15xxx configuration in System memory boot mode**

Feature/Peripheral	State	Comment
Clock source	HSI enabled	The system clock is equal to 16 MHz
USART1_RX pin	Input	PA10 pin: USART1 receives
USART1_TX pin	Output	PA9 pin: USART1 transmits
USART2_RX pin	Input	PD06 pin: USART2 receives
USART2_TX pin	Output	PD05 pin: USART2 transmits
SysTick timer	Enabled	Used to automatically detect the serial baud rate from the host.
USART1	Enabled	Once initialized the USART1 configuration is: 8-bits, even parity and 1 Stop bit
USART2	Enabled	Once initialized the USART2 configuration is: 8-bits, even parity and 1 Stop bit
RAM	-	2 Kbytes starting from address 0x2000 0000 are used by the bootloader firmware
System memory	-	4 Kbytes starting from address 0x1FF0 0000, contain the bootloader firmware
IWDG	-	The independent watchdog (IWDG) prescaler is configured to its maximum value and is periodically refreshed to prevent watchdog reset (in case the hardware IWDG option was previously enabled by the user)
Power	-	Voltage range is set to Voltage Range 2

The system clock is derived from the embedded internal high-speed RC, no external quartz is required for the bootloader code

After downloading the application binary, if you choose to execute the Go command, the peripheral registers used by the bootloader (shown in the above table) are not initialized to their default reset values before jumping to the user application. They should be reconfigured in the user application if they are used. So, if the IWDG is being used in the application, the IWDG prescaler value has to be adapted to meet the requirements of the application (since the prescaler was set to its maximum value by the bootloader).

## 7.2 Bootloader hardware requirements

The hardware required to put the STM32L15xx devices into System memory boot mode consists of any circuitry, switch or jumper, capable of holding the BOOT0 pin high and the BOOT1 pin low during reset.

To connect to the STM32L15xx devices during System memory boot mode, the following conditions have to be verified:

- The RX pins of the peripherals unused in this bootloader have to be kept at a known (low or high) level, and should not be left floating during the detection phase as described below:
  - If USART1 is used to connect to the bootloader: the USART2\_RX (PD6) pin has to be kept at a high or low level and must not be left floating during the detection phase.
  - If USART2 is used to connect to the bootloader: the USART1\_RX (PA10) pin has to be kept at a high or low level and must not be left floating during the detection phase.
- The peripheral to be used has to be connected through an RS232 serial interface (example, ST3232 RS232 transceiver) which must be:
  - Directly connected to the USART1\_RX (PA10) and USART1\_TX (PA9) pins when USART1 is used
  - Directly connected to the USART2\_RX (PD6) and USART2\_TX (PD5) pins when USART2 is used

The USART1\_CK, USART1\_CTS and USART1\_RTS pins are not used, therefore the application can use these pins for other peripherals or GPIOs. The same note is applicable for USART2.

The user can control the BOOT0 and Reset pins from a PC serial applet using the RS232 serial interface which controls BOOT0 through the CTS line and Reset through the DCD line. The user must use a full null modem cable. The necessary hardware to implement for this control exists in the STM32L152-EVAL board. For more details about this, refer to the “STM32L152-EVAL board user manual” (UM1009), available from the STMicroelectronics website: [www.st.com](http://www.st.com).

## 7.3 Bootloader selection

The STM32L15xx devices embedded bootloader supports two peripherals interfaces: USART1 and USART2. Any one of these peripheral interfaces can be used to communicate with the bootloader and download the application code to the internal Flash.

The embedded bootloader firmware is able to auto-detect the peripheral interface to be used. In an infinite loop, it detects any communication on the supported bootloader interfaces.

**Note:** *The RX pins of the peripherals not used in this bootloader must be kept at a known (low or high) level and should not be left floating during the detection phase as described below. Refer to [Section 7.2: Bootloader hardware requirements](#) for more information.*

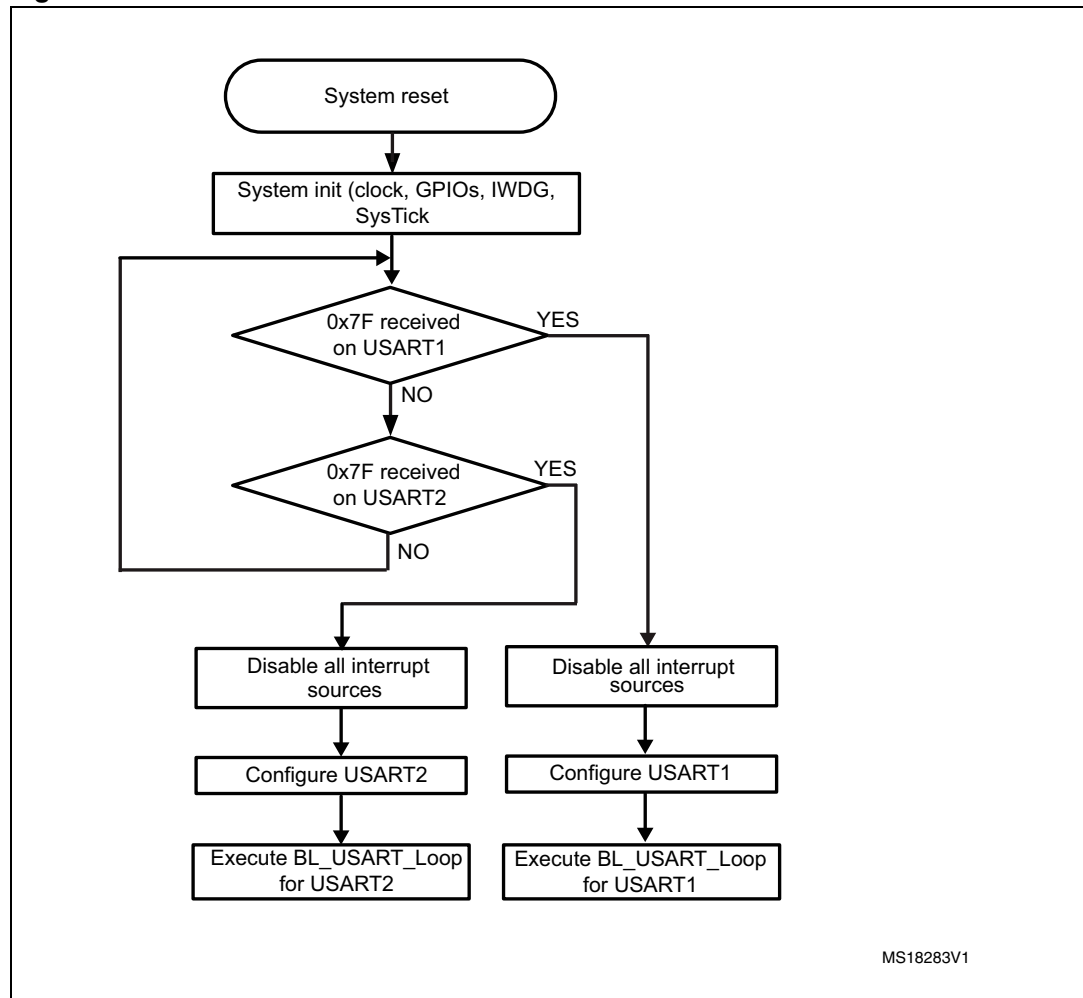
To use the USART bootloader on USART1 or USART2, connect the serial cable to the desired interface. Once the bootloader detects the data byte 0x7F on this interface, the

bootloader firmware executes the autobaudrate sequence and then enters a loop, waiting for any USART bootloader command.

Once one interface is selected for the bootloader, the other interface is disabled.

The figure below shows the bootloader detection mechanism. More details are provided in the sections corresponding to each peripheral bootloader.

**Figure 4. Bootloader selection for STM32L15xxx devices**



## 7.4 Important considerations

The bootloader of the Medium-density ultralow power devices has some specific features that should be taken into consideration, as described below:

- In addition to standard memories (internal Flash, internal SRAM, option bytes and System memory), the STM32L15xxx device bootloader firmware supports Data Memory (4 Kbytes from 0x08080000 to 0x08080FFF). Refer to the PM0062 Programming manual for more information.
- **Flash memory write** operations are performed through a program memory half page write operation. The bootloader firmware manages half page write operations at non-aligned addresses. Consequently, all write operations must only be Word-aligned (the address should be a multiple of 4). The number of data to be written must also be a multiple of 4 (non-aligned half page write addresses are accepted). Be aware of the duration needed for a write operation by referring to the product datasheet.
- **Data memory** can be read and written but cannot be erased using the Erase Command. When writing in a Data memory location, the bootloader firmware manages the erase operation of this location before any write. A write to Data memory must be Word-aligned (address to be written should be a multiple of 4) and the number of data must also be a multiple of 4. To erase a Data memory location, you can write zeros at this location.
- **Option byte**  
Address is 0x1FF80000. They allow three levels of protection:
  - Level 0
  - Level 1
  - Level 2Refer to PM0062 programming manual for more details about protection levels.
- **Read protect** command corresponds to the Level 1 protection.
- **Read unprotect** command corresponds to the Level 0 protection.
- **Mass erase** command is not supported by STM32L15xxx device Bootloader firmware. To perform a mass erase operation, two options are available:
  - Erase all sectors one by one using the Erase command
  - Set protection level to Level 1. Then, set it to Level 0 (using the Read protect command and then the Read Unprotect command). This operation results in a mass erase of the internal Flash memory (refer to Programming Manual PM0062 for more details).

## 7.5 Bootloader version

The following table lists the STM32L15xxx bootloader versions.

**Table 11. STM32L15xxx bootloader versions**

Bootloader version number	Description	Known limitations
V2.0	Initial bootloader version.	When a Read Memory command or Write Memory command is issued with an unsupported memory address and a correct address checksum (ie. address 0x6000 0000), the command is aborted by the bootloader device, but the NACK (0x1F) is not sent to the host. As a result, the next 2 bytes (which are the number of bytes to be read/written and its checksum) are considered as a new command and its checksum. <sup>(1)</sup>

1. If the “number of data - 1” (N-1) to be read/written is not equal to a valid command code (0x00, 0x01, 0x02, 0x11, 0x21, 0x31, 0x43, 0x44, 0x63, 0x73, 0x82 or 0x92), then the limitation is not perceived from the host since the command is NACKed anyway (as an unsupported new command).

## 8 STM32F205/215xx, and STM32F207/217xx bootloader

Through all this section **STM32F2xx** will be used as reference to STM32F205xx, STM32F207xx, STM32F215xx and STM32F217xx devices.

Two bootloader versions are available on STM32F2xx devices:

- V2.0 supporting USART1 and USART3  
This version is embedded in STM32F2xx devices revision B.
- V3.2 supporting USART1, USART3, CAN2 and DFU (USB FS Device)  
This version is embedded in STM32F2xx devices revision Y.

### 8.1 Bootloader V2.x

#### 8.1.1 Bootloader configuration

The bootloader V2.x embedded in STM32F2xx devices support two serial interfaces: USART1 and USART3 peripherals.

The following table shows the required hardware resources of STM32 devices used by the bootloader V2.x in System memory boot mode.

**Table 12. STM32F2xx configuration in System memory boot mode**

Feature/Peripheral	State	Comment
Clock source	HSI enabled	The system clock is equal to 24 MHz
USART1_RX pin	Input	PA10 pin: USART1 receives
USART1_TX pin	Output	PA9 pin: USART1 transmits
USART3_RX pin	Input	PC11 pin: USART3 receives
USART3_TX pin	Output	PC10pin: USART3 transmits
USART3_RX pin	Input	PB11 pin: USART3 receives
USART3_TX pin	Output	PB10pin: USART3 transmits
SysTick timer	Enabled	Used to automatically detect the serial baud rate from the host.
USART1	Enabled	Once initialized the USART1 configuration is: 8-bits, even parity and 1 Stop bit
USART3	Enabled	Once initialized the USART3 configuration is: 8-bits, even parity and 1 Stop bit
RAM	-	8 Kbytes starting from address 0x2000 0000
System memory	-	30688 bytes starting from address 0x1FFF 0000 contain the bootloader firmware

**Table 12. STM32F2xx configuration in System memory boot mode (continued)**

Feature/Peripheral	State	Comment
IWDG	-	The independent watchdog (IWDG) prescaler is configured to its maximum value and is periodically refreshed to prevent watchdog reset (in case the hardware IWDG option was previously enabled by the user)
Power	-	Voltage range is set to Voltage Range: [1.62V, 2.1V] (voltage range can be configured in run time using bootloader commands. Note that in this range internal Flash write operations are allowed only in byte format (Half-Word, Word and Double-Word operations are not allowed).

The system clock is derived from the embedded internal high-speed RC. No external quartz is required for the bootloader code.

After downloading the application binary, if you choose to execute the Go command, the peripheral registers used by the bootloader (shown in the above table) are not initialized to their default reset values before jumping to the user application. They should be reconfigured in the user application if they are used. So, if the IWDG is being used in the application, the IWDG prescaler value has to be adapted to meet the requirements of the application (since the prescaler was set to its maximum value by the bootloader).

### 8.1.2 Bootloader hardware requirements

The hardware required to put the STM32F2xx into System memory boot mode consists of any circuitry, switch or jumper, capable of holding the BOOT0 pin high and the BOOT1 pin low during reset.

To connect to the STM32F2xx during System memory boot mode, the following conditions have to be verified:

- The RX pins of the peripherals unused in this bootloader have to be kept at a known (low or high) level, and should not be left floating during the detection phase as described below:
  - If USART1 is used to connect to the bootloader: the USART3\_RX (PC11 and PB11) pins have to be kept at a high or low level and must not be left floating during the detection phase.
  - If USART3 (on PB10/PB11) is used to connect to the bootloader: the USART1\_RX (PA10) and the other USART3\_RX pin (PC11) have to be kept at a high or low level and must not be left floating during the detection phase.
  - If USART3 (on PC10/PC11) is used to connect to the bootloader: the USART1\_RX (PA10) and the other USART3\_RX pin (PB11) have to be kept at a high or low level and must not be left floating during the detection phase.
- Connect to the peripheral to be used through:
  - An RS232 serial interface (example, ST3232 RS232 transceiver) has to be directly connected to the USART1\_RX (PA10) and USART1\_TX (PA9) pins when USART1 is used, or to the USART3\_RX (PB11 or PC11) and USART3\_TX (PB10 or PC10) pins or when USART3 is used.

The USART1\_CK, USART1\_CTS and USART1\_RTS pins are not used, therefore the application can use these pins for other peripherals or GPIOs. The same note is applicable for USART3.



The user can control the BOOT0 and Reset pins from a PC serial applet using the RS232 serial interface which controls BOOT0 through the CTS line and Reset through the DCD line. The user must use a full null modem cable. The necessary hardware to implement for this control exists in the STM322xG-EVAL board. For more details, refer to the STM322xG-EVAL board user manual, available from the STMicroelectronics website: [www.st.com](http://www.st.com).

### 8.1.3 Bootloader selection

The STM32F2xx embedded bootloader V2.x supports two peripheral interfaces: USART1 and USART3 (on PB10/PB11 and PC10/PC11). Any one of these peripheral interfaces can be used to communicate with the bootloader and download the application code to the internal Flash memory.

The embedded bootloader firmware is able to auto-detect the peripheral interface to be used. In an infinite loop, it detects any communication on the supported bootloader interfaces.

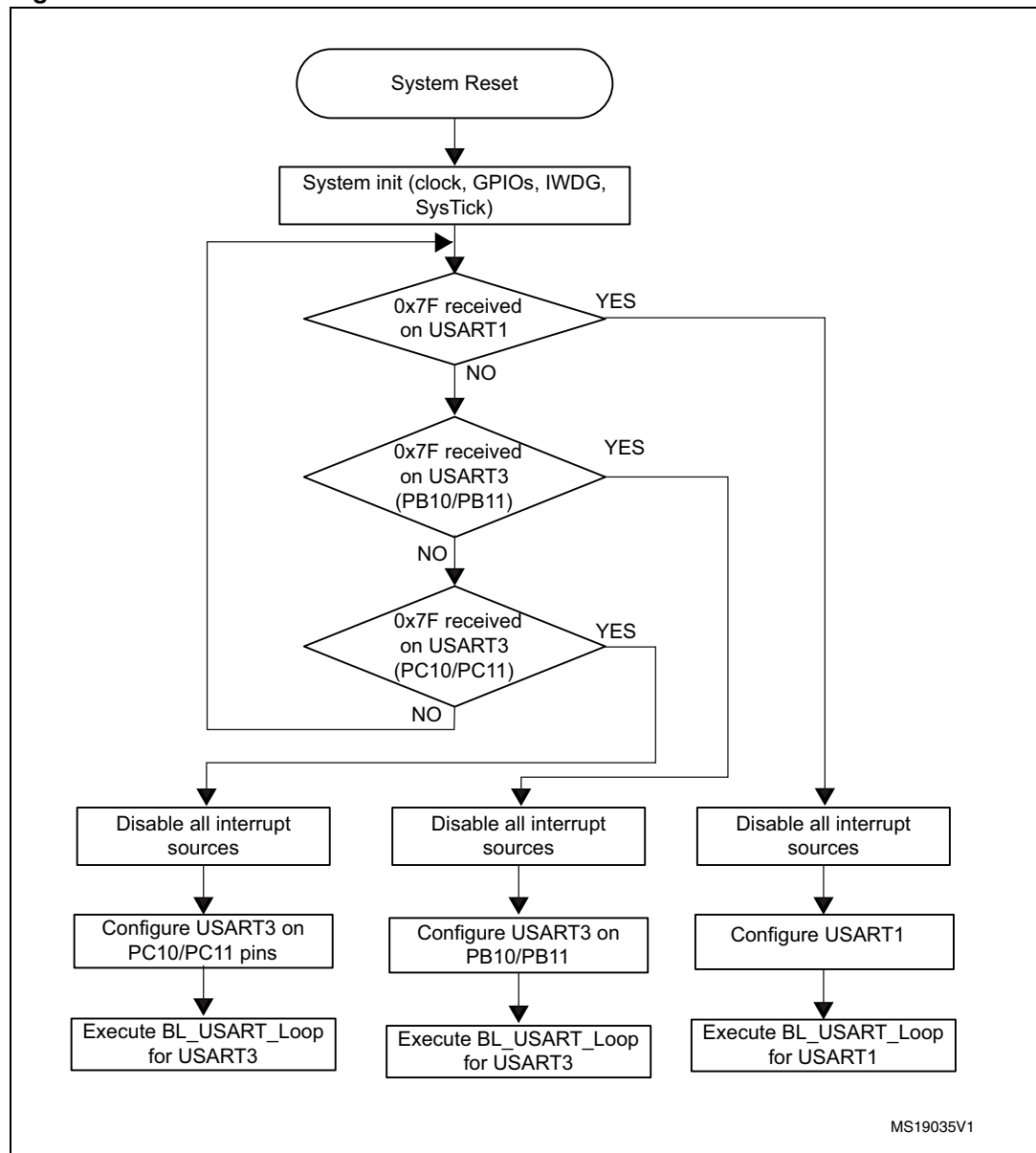
**Note:** *The RX pins of the peripherals not used in this bootloader must be kept at a known (low or high) level and should not be left floating during the detection phase as described below. Refer to [Section 8.1.2: Bootloader hardware requirements](#) for more information.*

To use the USART bootloader on USART1 or USART3, connect the serial cable to the desired interface. Once the bootloader detects the data byte 0x7F on this interface, the bootloader firmware executes the autobaudrate detection sequence and enters a loop, waiting for any USART bootloader command.

Once an interface is selected for the bootloader, the other interfaces are disabled.

[Figure 5](#) shows the bootloader detection mechanism. More details are provided in the sections corresponding to each peripheral bootloader.

Figure 5. Bootloader V2.x selection for STM32F2xx



### 8.1.4 Important considerations

The STM32F2xx bootloader has some specific features that should be taken into consideration:

- In addition to standard memories (internal Flash, internal SRAM, option bytes and System memory), STM32F2xx bootloader firmware supports OTP memory (512 bytes from 0x1FFF 7800 to 0x1FFF 7A00). Refer to PM0059 Flash programming manual for more information.
- **OTP memory** can be read and written but cannot be erased using the Erase command. When writing in an OTP memory location, make sure that the relative protection bit (in the last 16 bytes of the OTP memory) is not reset.

- **Option bytes**

Address is 0x1FFFC000. They allow three levels of protection:

- Level 0
- Level 1
- Level 2

Refer to PM0059 programming manual for more details about protection levels.

- **Read protect** command corresponds to Level 1 protection.
- **Read unprotect** command corresponds to Level 0 protection.
- **Mass erase** command on STM32F2xx takes longer than on other STM32F devices due to their memory density. Make sure that the timeout used by your host interface to wait for an acknowledge event after sending a Mass erase command is sufficient.
- **Voltage Range configuration**

The Voltage Range can be updated on the fly by the bootloader software. The Voltage Range is set to its default value at each bootloader software startup (after system reset or jump to the bootloader code). The bootloader software allows modifying this parameter through a virtual memory location. This memory location is not physical but can be read and written using usual bootloader read/write operations according to the protocol in use (USART, CAN or DFU). This memory location contains 4 bytes which are described in [Table 13](#). It can be accessed by 1, 2, 3 or 4 bytes. However, reserved bytes should remain at their default values (0xFF), otherwise the request will be NACKed.

**Table 13. STM32F2xx Voltage Range configuration using bootloader V2.x**

Address	Size	Description
0xFFFF0000	1 byte	This byte controls the current value of Voltage Range: 0x00: Voltage Range [1.62V, 2.1V] 0x01: Voltage Range [2.1V, 2.4V] 0x02: Voltage Range [2.4V, 2.7V] 0x03: Voltage Range [2.7V, 3.6V] 0x04: Voltage Range [2.7V, 3.6V] and Double Word write/erase operation is used. In this case it is mandatory to supply 9 V through VPP pin (refer to PM0059 for more details about Double-Word write operation). Other: all other values are not supported and will be NACKed.
0xFFFF0001	1 byte	Reserved. 0xFF: Default value. Other: all other values are not supported and will be NACKed.
0xFFFF0002	1 byte	Reserved. 0xFF: Default value. Other: all other values are not supported and will be NACKed.
0xFFFF0003	1 byte	Reserved. 0xFF: Default value. Other: all other values are not supported and will be NACKed.

### 8.1.5 Bootloader V2.x versions

[Table 13](#) lists the V2.x versions of STM32F2xx bootloader.

**Table 14. STM32F2xx bootloader V2.x versions**

Bootloader version number	Description	Known limitations
V2.0	Initial V2.x bootloader version.	When a Read Memory command or Write Memory command is issued with an unsupported memory address and a correct address checksum (ie. address 0x6000 0000), the command is aborted by the bootloader device, but the NACK (0x1F) is not sent to the host. As a result, the next 2 bytes (which are the number of bytes to be read/written and its checksum) are considered as a new command and its checksum. <sup>(1)</sup>

1. If the "number of data - 1" (N-1) to be read/written is not equal to a valid command code (0x00, 0x01, 0x02, 0x11, 0x21, 0x31, 0x43, 0x44, 0x63, 0x73, 0x82 or 0x92), then the limitation is not perceived from the host since the command is NACKed anyway (as an unsupported new command).

## 8.2 Bootloader V3.x

### 8.2.1 Bootloader configuration

The bootloader V3.x embedded in STM32F2xx devices support four serial peripherals: USART1, USART3, CAN2, and DFU (USB FS Device).

[Table 15](#) shows the required hardware resources of STM32F2xx devices used by the bootloader V3.x in System memory boot mode.

**Table 15. STM32F2xx configuration in System memory boot mode**

Bootloader	Feature/Peripheral	State	Comment
Common to all bootloaders	RCC	HSI enabled	The system clock is equal to 24 MHz using the PLL. The HSI clock source is used at startup (interface detection phase) and when USARTx interfaces are selected (once CAN or DFU bootloader is selected, the clock source will be derived from external crystal).
		HSE enabled	The system clock is equal to 60 MHz. The HSE clock source is used only when the CAN or the DFU (USB FS Device) interfaces are selected. The external clock must provide a frequency multiple of 1 MHz and ranging from 4 MHz to 26 MHz.
		-	The Clock Security System (CSS) interrupt is enabled for the CAN and DFU bootloaders. Any failure (or removal) of the external clock generates system reset.
	RAM	-	8 Kbytes starting from address 0x2000 0000 are used by the bootloader firmware
	System memory	-	30688 bytes starting from address 0x1FF0 0000, contain the bootloader firmware
	IWDG	-	The independent watchdog (IWDG) prescaler is configured to its maximum value. It is periodically refreshed to prevent watchdog reset (in case the hardware IWDG option was previously enabled by the user).
	Power	-	Voltage range is set to [1.62V, 2.1V]. The voltage range can be configured in run time using bootloader commands. Note that in this range internal Flash write operations are allowed only in byte format (Half-Word, Word and Double-Word operations are not allowed).

Table 15. STM32F2xx configuration in System memory boot mode (continued)

Bootloader	Feature/Peripheral	State	Comment
USART1 Bootloader	USART1	Enabled	Once initialized the USART1 configuration is: 8-bits, even parity and 1 Stop bit
	USART1_RX pin	Input	PA10 pin: USART1 in reception mode
	USART1_TX pin	Output	PA9 pin: USART1 in transmission mode
	USART3_RX (PB11), USART3_RX (PC11), CAN2_RX (PB05), OTG_FS_DM (PA11) and OTG_FS_DP (PA12) pins must be kept at a high or low level during the detection phase.		
USART3 Bootloader (on PB10/PB11)	USART3	Enabled	Once initialized the USART3 configuration is: 8-bits, even parity and 1 Stop bit
	USART3_RX pin	Input	PB11 pin: USART3 in reception mode
	USART3_TX pin	Output	PB10pin: USART3 in transmission mode
	USART1_RX (PA10), USART3_RX (PC11), CAN2_RX (PB05), OTG_FS_DM (PA11) and OTG_FS_DP (PA12) pins must be kept at a high or low level during the detection phase.		
USART3 Bootloader (on PC10/PC11)	USART3	Enabled	Once initialized the USART3 configuration is: 8-bits, even parity and 1 Stop bit
	USART3_RX pin	Input	PC11 pin: USART3 in reception mode
	USART3_TX pin	Output	PC10pin: USART3 in transmission mode
	USART1_RX (PA10), USART3_RX (PB11), CAN2_RX (PB05), OTG_FS_DM (PA11) and OTG_FS_DP (PA12) pins must be kept at a high or low level during the detection phase.		
USART1 and USART3 Bootloaders	SysTick timer	Enabled	Used to automatically detect the serial baud rate from the host for USARTx bootloaders.
CAN2 bootloader	CAN2	Enabled	Once initialized the CAN2 configuration is: Baudrate 125 kbps, 11-bit identifier. <b>Note:</b> CAN1 is clocked during CAN2 bootloader execution because STM32F2xx CAN1 manages the communication between CAN2 and SRAM.
	CAN2_RX pin	Input	PB05 pin: CAN2 in reception mode
	CAN2_TX pin	Output	PB13pin: CAN2 in transmission mode
	USART1_RX (PA10), USART3_RX (PB11), USART3_RX (PC11), OTG_FS_DM (PA11) and OTG_FS_DP (PA12) pins must be kept at a high or low level during the detection phase.		
DFU bootloader	USB_OTG_FS	Enabled	USB OTG FS configured in Forced Device mode. USB_OTG_FS interrupt vector is enabled and used for USB DFU communications.
	USB_OTG_FS_DM pin	Input	PA11 pin: USB OTG FS DM line
	USB_OTG_FS_DP pin	Output	PA12pin: USB OTG FS DP line
	USART1_RX (PA10), USART3_RX (PB11), USART3_RX (PC11) and CAN2_RX (PB05) pins must be kept at a high or low level during the detection phase.		
CAN2 and DFU bootloaders	TIM11	Enabled	This timer is used to determine the value of the external clock frequency. Once the external clock frequency is determined, the RCC system is configured to operate at 60 MHz system clock (using PLL).

The system clock is derived from the embedded internal high-speed RC for USARTx bootloaders. No external quartz is required in this case for the bootloader code. This internal clock is also used for CAN and DFU (USB FS Device) but only for the selection phase. An external clock multiple of 1 MHz (between 4 and 26 MHz) is required for CAN and DFU bootloader execution after the selection phase.

The CAN and DFU bootloaders implement an external clock detection mechanism allowing to determine the value of the external clock using the internal high-speed RC and TIM11 timer. The accuracy of this mechanism allows to detect only frequencies multiple of 1 MHz and ranging from 4 to 26 MHz. Any other value is not supported and will result in unexpected behavior of the bootloader.

After downloading the application binary, if you choose to execute the Go command, the peripheral registers used by the bootloader (shown in the above table) are not initialized to their default reset values before jumping to the user application. They should be reconfigured in the user application if they are used. So, if the IWDG is being used in the application, the IWDG prescaler value has to be adapted to meet the requirements of the application (since the prescaler was set to its maximum value by the bootloader).

## 8.2.2 Bootloader hardware requirements

The hardware required to put the STM32F2xx into System memory boot mode consists of any circuitry, switch or jumper, capable of holding the BOOT0 pin high and the BOOT1 pin low during reset.

To connect to the STM32F2xx during System memory boot mode, the following conditions have to be verified:

- The RX pins of the peripheral unused in this bootloader have to be kept at a known (low or high) level, and should not be left floating during the detection phase as described below:
  - If USART1 is used to connect to the bootloader: the USART3\_RX (PC11 and PB11), CAN2\_RX (PB05), OTG\_FS\_DM (PA11) and OTG\_FS\_DP (PA12) pins have to be kept at a high or low level and must not be left floating during the detection phase.
  - If USART3 (on PB10/PB11) is used to connect to the bootloader: the USART1\_RX (PA10), USART3\_RX (PC11), CAN2\_RX (PB05), OTG\_FS\_DM (PA11) and OTG\_FS\_DP (PA12) have to be kept at a high or low level and must not be left floating during the detection phase.
  - If USART3 (on PC10/PC11) is used to connect to the bootloader: the USART1\_RX (PA10), USART3\_RX pin (PB11), CAN2\_RX (PB05), OTG\_FS\_DM (PA11) and OTG\_FS\_DP (PA12) have to be kept at a high or low level and must not be left floating during the detection phase.
  - If CAN2 is used to connect to the bootloader: the USART1\_RX (PA10), USART3\_RX (PC11 and PB11), OTG\_FS\_DM (PA11) and OTG\_FS\_DP (PA12) pins have to be kept at a high or low level and must not be left floating during the detection phase.
  - If DFU (USB FS Device) is used to connect to the bootloader: the USART1\_RX (PA10), USART3\_RX (PC11 and PB11) and CAN2\_RX (PB05) pins have to be kept at a high or low level and must not be left floating during the detection phase.

- Connect to the peripheral to be used through:
  - An RS232 serial interface (example, ST3232 RS232 transceiver) has to be directly connected to the USART1\_RX (PA10) and USART1\_TX (PA9) pins when USART1 is used, or to the USART2\_RX (PD6) and USART2\_TX (PD5) pins when USART2 is used
  - A CAN interface (CAN transceiver) has to be directly connected to the CAN2\_RX (PB5) and CAN2\_TX (PB13) pins.
  - A certified USB cable has to be connected to the microcontroller (optionally an ESD protection circuitry can be used).

The USART1\_CK, USART1\_CTS and USART1\_RTS pins are not used, therefore the application can use these pins for other peripherals or GPIOs. The same note is applicable for USART3.

The user can control the BOOT0 and Reset pins from a PC serial applet using the RS232 serial interface which controls BOOT0 through the CTS line and Reset through the DCD line. The user must use a full null modem cable. The necessary hardware to implement for this control exists in the STM322xG-EVAL board. For more details about this, refer to the STM322xG-EVAL board user manual, available from the STMicroelectronics website: [www.st.com](http://www.st.com).

### 8.2.3 Bootloader selection

The STM32F2xx embedded bootloader V3.x supports three peripheral interfaces: USART1, USART3 (on PB10/PB11 and PC10/PC11), CAN2 and DFU (USB FS Device). Any one of these peripheral interfaces can be used to communicate with the bootloader and download the application code to the internal Flash.

The embedded bootloader firmware is able to auto-detect the peripheral interface to be used. In an infinite loop, it detects any communication on the supported bootloader interfaces.

**Note:** *The RX pins of the peripherals not used in this bootloader must be kept at a known (low or high) level and should not be left floating during the detection phase as described below. Refer to [Section 8.2.2: Bootloader hardware requirements](#) for more information.*

To use the USART bootloader on USART1 or USART3, connect the serial cable to the desired interface. Once the bootloader detects the data byte 0x7F on this interface, the bootloader firmware executes the auto-baud rate sequence and then enters a loop, waiting for any USART bootloader command.

To use the CAN2 interface, connect the CAN cable to CAN2. Once the bootloader detects a frame on the CAN2\_RX pin (PB5), the bootloader firmware enters a CAN loop and starts to determine the external clock frequency value. The supported HSE frequencies are multiple of 1 MHz ranging from 4 to 26 MHz. Any other values leads to an unexpected behavior, CAN bootloader firmware enters an infinite loop and waits until it receives a message. If the external clock is not present, a system reset is generated.

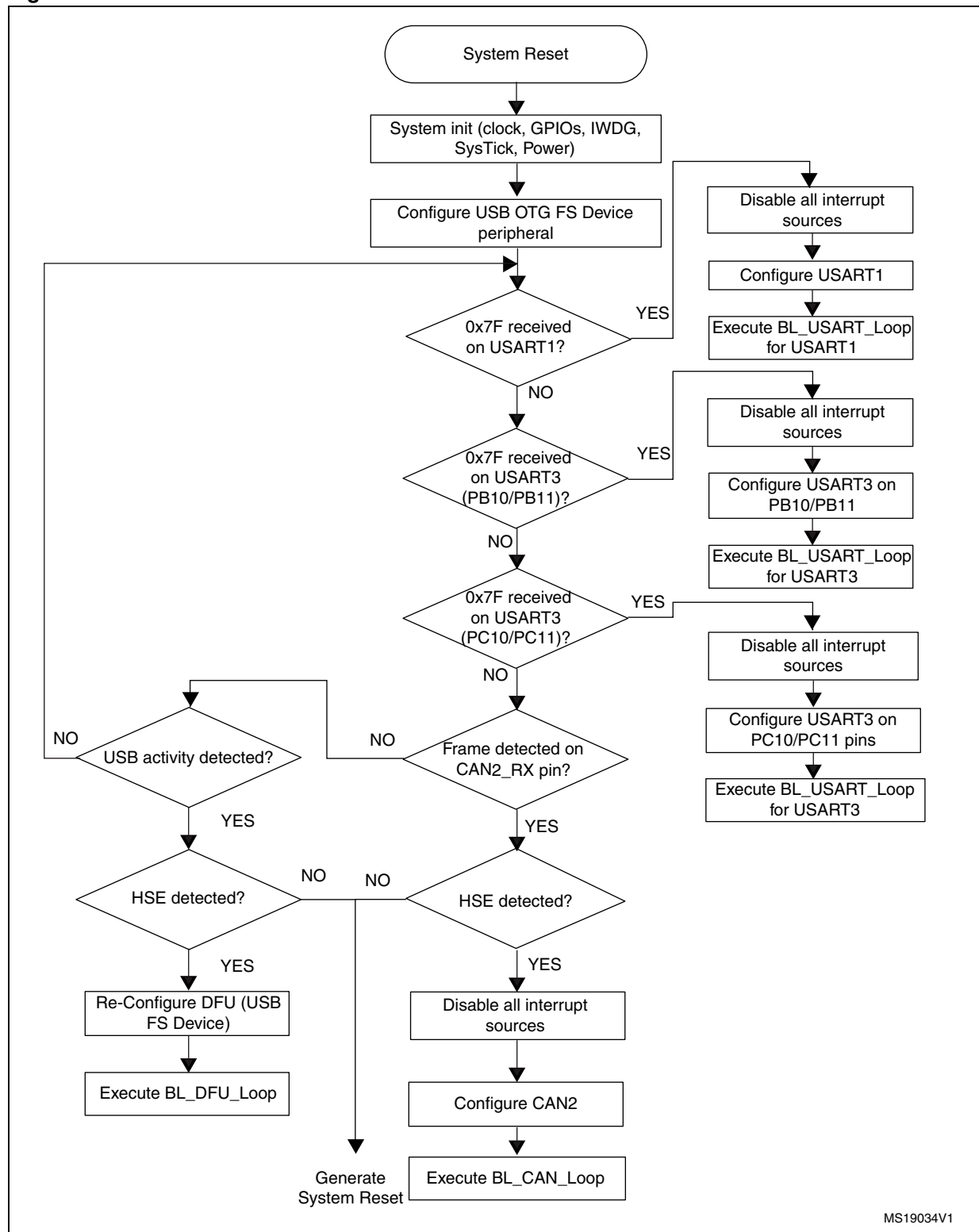
If a USB cable is plugged into the microcontroller's USB interface at any time during the bootloader firmware selection sequence, the bootloader enters the DFU bootloader loop waiting for any DFU bootloader command.



To use the USART or the CAN bootloader, it is mandatory that no USB Host is connected to the USB peripheral during the selection phase. Once the USART or CAN bootloader is selected, the user can plug a USB cable without impacting the selected bootloader execution except commands which generate a system reset.

Once one interface is selected for the bootloader, all other interfaces are disabled. The figure below shows the bootloader selection mechanism. More details are provided in the sections corresponding to each peripheral bootloader.

Figure 6. Bootloader V3.x selection for STM32F2xx



## 8.2.4 Important considerations

STM32F2xx bootloader has some specific features that should be taken into consideration:

- In addition to standard memories (internal Flash, internal SRAM, option bytes and System memory), STM32F2xx devices bootloader firmware supports OTP memory (512 bytes from 0x1FFF 7800 to 0x1FFF 7A00, refer to PM0059 programming manual for more information).
- **OTP memory** can be read and written but cannot be erased using Erase command. When writing in an OTP memory location, make sure that the relative protection bit (in the last 16 bytes of the OTP memory) is not reset.
- **Option bytes**  
Address is 0x1FFFC000. They allow three levels of protection:
  - Level 0
  - Level 1
  - Level 2

Refer to PM0059 programming manual for more details about protection levels.

- **Read protect** command corresponds to Level 1 protection.
- **Read unprotect** command corresponds to Level 0 protection.
- **Mass erase** command on STM32F2xx takes longer than on other STM32 devices due to their memory density. Make sure that the timeout used by your host interface to wait for an acknowledge event after sending a Mass erase command is sufficient.
- **Voltage Range configuration**  
The Voltage Range can be updated on the fly by the bootloader software. The Voltage Range is set to its default value at each bootloader software startup (after system reset or jump to the bootloader code). The bootloader software allows modifying this parameter through a virtual memory location. This memory location is not physical but can be read and written using usual bootloader read/write operations according to the protocol in use (USART, CAN or DFU). This memory location contains 4 bytes which are described in [Table 16](#). It can be accessed by 1, 2, 3 or 4 bytes. However, reserved bytes should remain at their default values (0xFF), otherwise the request will be NACKed.

**Table 16. STM32F2xx Voltage Range configuration using bootloader V3.x**

Address	Size	Description
0xFFFF0000	1 byte	This byte controls the current value of Voltage Range: 0x00: Voltage Range [1.62V, 2.1V] 0x01: Voltage Range [2.1V, 2.4V] 0x02: Voltage Range [2.4V, 2.7V] 0x03: Voltage Range [2.7V, 3.6V] 0x04: Voltage Range [2.7V, 3.6V] and Double Word write/erase operation is used. In this case it is mandatory to supply 9 V through VPP pin (refer to PM0059 for more details about Double-Word write procedure). Other: All other values are not supported and will be NACKed.
0xFFFF0001	1 byte	Reserved. 0xFF: Default value. Other: all other values are not supported and will be NACKed.
0xFFFF0002	1 byte	Reserved. 0xFF: Default value. Other: all other values are not supported and will be NACKed.
0xFFFF0003	1 byte	Reserved. 0xFF: Default value. Other: all other values are not supported and will be NACKed.

## 8.2.5 Bootloader version V3.x

[Table 16](#) lists the V3.x versions of STM32F2xx bootloader.

**Table 17. STM32F2xx bootloader V3.x versions**

Bootloader version number	Description	Known limitations
V3.2	Initial V3.x bootloader version.	When a Read Memory command or Write Memory command is issued with an unsupported memory address and a correct address checksum (ie. address 0x6000 0000), the command is aborted by the bootloader device, but the NACK (0x1F) is not sent to the host. As a result, the next 2 bytes (which are the number of bytes to be read/written and its checksum) are considered as a new command and its checksum. <sup>(1)</sup>

1. If the “number of data - 1” (N-1) to be read/written is not equal to a valid command code (0x00, 0x01, 0x02, 0x11, 0x21, 0x31, 0x43, 0x44, 0x63, 0x73, 0x82 or 0x92), then the limitation is not perceived from the host since the command is NACKed anyway (as an unsupported new command).

## 9 Device-dependent bootloader parameters

The bootloader protocol's command set and sequences for each serial peripheral (USART, CAN and USB) are the same for all STM32 devices. Some parameters, however, are device-dependent. For a few commands, the value of some parameters may depend on the device used. These parameters are listed below:

- PID (product ID), which changes with the device
- Valid memory addresses (RAM, Flash memory, System memory, option byte area) accepted by the bootloader when the Read Memory, Go and Write Memory commands are accepted.
- Size of the Flash memory sector used when executing the Write Protect command.

The table below shows the values of these parameters for each STM32 device bootloader in production.

**Table 18. Bootloader device-dependant parameters**

Device	Product (device) ID	RAM memory	Flash memory	Flash sector size	Option byte area	System memory
Low-density	0x412	0x20000200 up to 0x20002800	0x08000000 up to 0x08008000	4 Kbytes (4 pages of 1 Kbyte each)	0x1FFFF800 - 0x1FFFF80F	0x1FFF0000 - 0x1FFF8000
Medium-density	0x410	0x20000200 up to 0x20005000	0x08000000 up to 0x08020000	4 Kbytes (4 pages of 1 Kbyte each)	0x1FFFF800 - 0x1FFFF80F	0x1FFF0000 - 0x1FFF8000
High-density	0x414	0x20000200 up to 0x20010000	0x08000000 up to 0x08080000	4 Kbytes (2 pages of 2 Kbytes each)	0x1FFFF800 - 0x1FFFF80F	0x1FFF0000 - 0x1FFF8000
Connectivity line	0x418	0x20001000 up to 0x20010000	0x08000000 up to 0x08040000	4 Kbytes (2 pages of 2 Kbytes each)	0x1FFFF800 - 0x1FFFF80F	0x1FFFB000 - 0x1FFFF800
Medium-density value line	0x420	0x20000200 up to 0x20002000	0x08000000 up to 0x08020000	4 Kbytes (4 pages of 1 Kbyte each)	0x1FFFF800 - 0x1FFFF80F	0x1FFF0000 - 0x1FFF8000
High-density value line	0x428	0x20000200 up to 0x20008000	0x08000000 up to 0x08080000	4 Kbytes (2 pages of 2 Kbytes each)	0x1FFFF800 - 0x1FFFF80F	0x1FFF0000 - 0x1FFF8000
XL-density	0x430	0x20000800 up to 0x20018000	0x08000000 up to 0x08100000	4 Kbytes (2 pages of 2 Kbytes each)	0x1FFFF800 - 0x1FFFF80F	0x1FFFE000 - 0x1FFFF800
Medium-density ultralow power line	0x416	0x20000800 up to 0x20004000	0x08000000 up to 0x08020000	4 Kbytes (16 pages of 256 Bytes each)	0x1FF80000 - 0x1FF8000F	0x1FF00000 - 0x1FF01000
STM32F2xx devices	0x411	0x20002000 up to 0x20020000	0x08000000 up to 0x08100000	12 sectors (4x16Kbytes, 1x64Kbytes, 7x128Kbytes)	0x1FFFC000 - 0x1FFFC00F	0x1FFF0000 - 0x1FFF77DF

## 10 Bootloader timing characteristics

This section presents the main startup timings of the bootloader firmware depending on products. They can be used to set up the connection timeout, that is how long the host waits before synchronization with the bootloader is established.

Three types of timings will be described herein:

- Hardware-dependent timings relative to product and directly extracted from the product datasheet.
- Communication-dependent timings relative to the baudrate and data traffic on the bus. These timings depend only on the communication interface configuration and on the host behavior.
- Bootloader software-dependent timings relative to bootloader software operations.

All the timings described in this section are expressed in milliseconds (ms) except when otherwise specified.

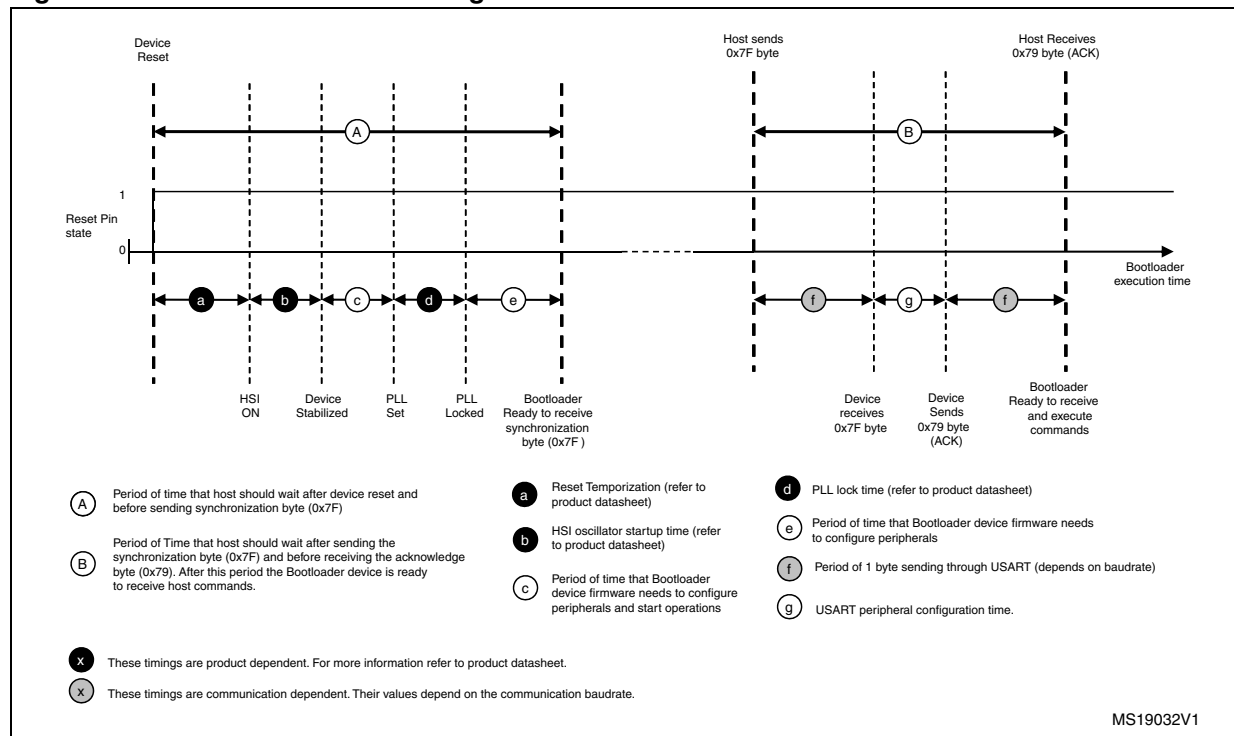
### 10.1 USART bootloader timing characteristics

Two main timings need to be considered for host operations when using the USART bootloader:

- Timing A  
After bootloader reset, this timing corresponds to the time during which the host waits before sending the synchronization data (0x7F) to properly configure the bootloader baudrate detection. This timing will be referred to as **A** throughout this section.
- Timing B  
After sending the synchronization data (0x7F), this timing corresponds to the time during which the host waits before receiving the first acknowledge response (meaning that the bootloader is ready to receive and execute host commands). This timing will be referred to as **B** throughout this section.

**A** and **B** timings are composed of different sub-timings as described in [Figure 7](#).

Figure 7. USART bootloader timing waveforms



The timing values for each product are listed in [Table 19](#), [Table 20](#), [Table 21](#), [Table 22](#), [Table 23](#), and [Table 24](#).

**Table 19. USART bootloader timings for low/medium/high-density and value line devices**

Time	Description	Min	Max	Unit
a	Reset temporization	1	4.5	ms
b	HSI oscillator startup time	0.001	0.002	ms
c	Bootloader firmware operations	0.004	-	ms
d	PLL Lock time	0.2	-	ms
e	Bootloader firmware operations	0.002	-	ms
f	One USART byte sending period	0.078125	7.5	ms
g	Bootloader firmware operations	0.002	-	ms
A	Time = a + b + c + d + e	1.207	4.708	ms
B	Time = (2 x f) + g	0.15825	15.002	ms

**Table 20. USART bootloader timings for XL-density line devices**

Time	Description	Min	Max	Unit
a	Reset temporization	1	4.5	ms
b	HSI oscillator startup time	0.001	0.002	ms
c	Bootloader firmware operations	0.02	-	ms
d	PLL Lock time	0.2	-	ms
e	Bootloader firmware operations	0.006	-	ms
f	One USART byte sending period	0.078125	7.5	ms
g	Bootloader firmware operations	0.006	-	ms
A	Time = a + b + c + d + e	1.227	4.728	ms
B	Time = (2 x f) + g	0.16225	15.006	ms



**Table 21. USART bootloader timings for connectivity line devices (PA9 pin low)**

Time	Description	Min	Max	Unit
a	Reset temporization	1	4.5	ms
b	HSI oscillator startup time	0.001	0.002	ms
c	Bootloader firmware operations	0.025	-	ms
d	PLL Lock time	0.35	-	ms
e	Bootloader firmware operations	0.02	-	ms
f	One USART byte sending period	0.078125	7.5	ms
g	Bootloader firmware operations	0.007	-	ms
A	Time = a + b + c + d + e	1.396	4.897	ms
B	Time = (2 x f) + g	0.16325	15.007	ms

For connectivity line devices, PA9 pin (USB\_VBUS) is used to detect the USB host connection. The initialization of USB peripheral is performed only if PA9 is high at detection phase which means that a host is connected to the port and delivering 5 V on the USB bus. When PA9 level is high at detection phase, more time is required to initialize and shutdown the USB peripheral.

To minimize bootloader detection time for connectivity line devices when PA9 pin is not used, keep PA9 state low during detection phase from the moment the device is reset till a device ACK is sent.

**Table 22. USART bootloader timings for connectivity line devices (PA9 high)**

Time	Description	Min	Max	Unit
a	Reset temporization	1	4.5	ms
b	HSI oscillator startup time	0.001	0.002	ms
c	Bootloader firmware operations	0.025	-	ms
d	PLL Lock time	0.35	-	ms
e	Bootloader firmware operations	523	-	ms
f	One USART byte sending period	0.078125	7.5	ms
g	Bootloader firmware operations	105	-	ms
A	Time = a + b + c + d + e	524.376	527.877	ms
B	Time = (2 x f) + g	105.1563	120	ms

**Table 23. USART bootloader timings for STM32L15xx medium-density ultralow power devices**

Time	Description	Min	Max	Unit
a	Reset temporization	0.4	1.6	ms
b	MSI oscillator stabilization time	-	0.04	ms
c	Bootloader firmware operations	0.064	-	ms
d	HSI oscillator startup time	0.0037	0.006	ms
e	Bootloader firmware operations	0.034	-	ms
f	One USART byte sending period	0.078125	7.5	ms
g	Bootloader firmware operations	0.008	-	ms
A	Time = a + b + c + d + e	0.5417	1.744	ms
B	Time = (2 x f) + g	0.16425	15.008	ms

**Table 24. USART bootloader timings for STM32F205/215xx and STM32F207/217xx devices**

Time	Description	Min	Max	Unit
a	Reset temporization	0.5	3.0	ms
b	HSI oscillator startup time	0.0022	0.004	ms
c	Bootloader firmware operations	0.01	-	ms
d	PLL Lock time	0.075	0.2	ms
e	Bootloader firmware operations	84	-	ms
f	One USART byte sending period	0.078125	7.5	ms
g	Bootloader firmware operations	0.009	-	ms
A	Time = a + b + c + d + e	84.5872	87.214	ms
B	Time = (2 x f) + g	0.16525	15.009	ms

## 10.2 USB bootloader timing characteristics

The main timings that need to be considered for host operations when using the USB bootloader are the following:

- Timing A

After bootloader reset, this timing corresponds to the time during which the host waits before starting the connection sequence with the device. It is similar to the USART connection timeout described in [Section 10.1](#). It will be referred to as **A** throughout this section.

- Timing B

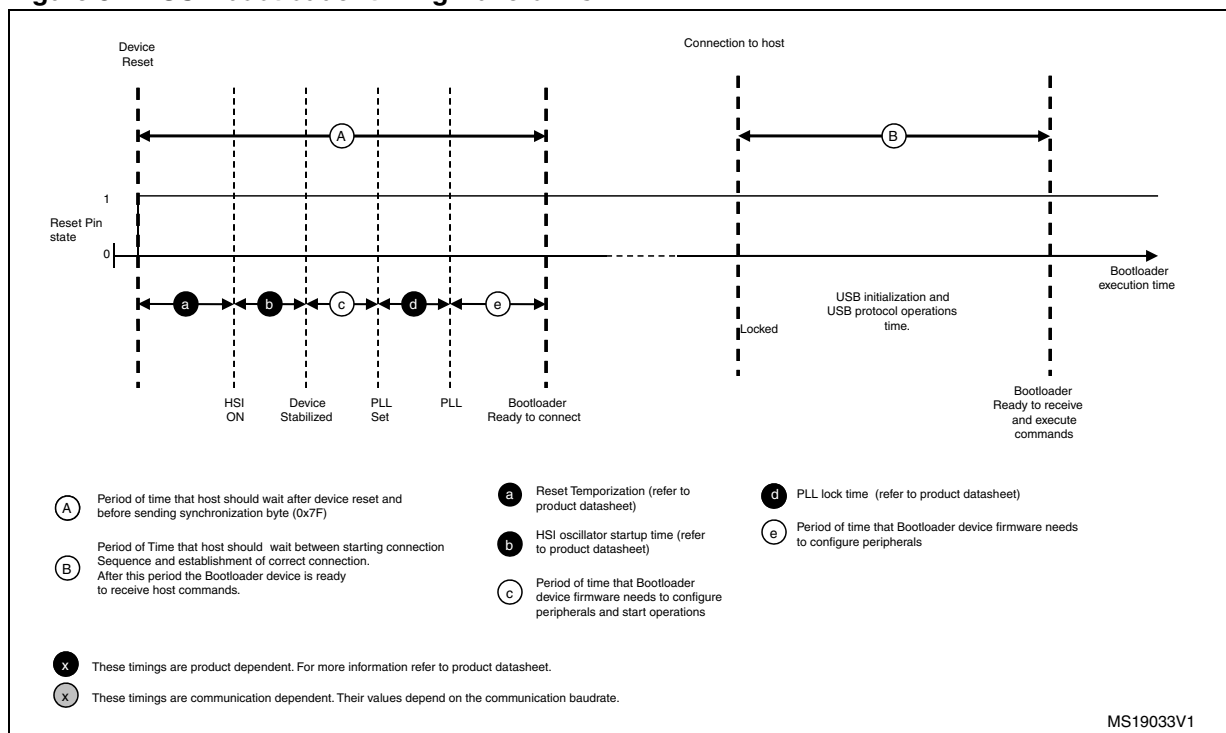
When the connection sequence has started, this timing corresponds to the time required by the device to establish a correct connection with the host (meaning that the bootloader is ready to receive and execute host commands). This timing includes enumerations and DFU components configuration (e.g. internal Flash memory). This timing will be referred to as **B** throughout this section.

For connectivity line devices, if the external HSE crystal frequency is different from 25 MHz (14.7456 MHz or 8 MHz), the device performs several unsuccessful enumerations (with connect – disconnect sequences) before being able to establish a correct connection with the host. This is due to the HSE automatic detection mechanism based on SOF detection.

**A** and **B** timings are composed of different sub-timings as described in [Figure 8](#). Refer to [Table 19](#), [Table 20](#), [Table 21](#), [Table 22](#), [Table 23](#), and [Table 24](#) for the values of timing A (identical to USART bootloader), and to [Table 25](#) and [Table 26](#) for the values of timing B.

**Note:** *For USB interface, only minimum timings are provided since the connection timing depends on environment and host configuration (number of nodes (hubs), host speed, traffic on the USB bus, host loading ...).*

**Figure 8. USB bootloader timing waveforms**



**Table 25. USB minimum timings for connectivity line devices**

Time	Description	25MHz	14.7456MHz	8MHz	Unit
a	Reset temporization	1	1	1	ms
b	HSI oscillator startup time	0.001	0.001	0.001	ms
c	Bootloader firmware operations	0.025	0.025	0.025	ms
d	PLL Lock time	0.35	0.35	0.35	ms
e	Bootloader firmware operations	523	523	523	ms
A	Time = a + b + c + d + e	524.376	524.376	524.376	ms
B	Connection establishment	460	4500	13700	ms

**Table 26. USB minimum timings for STM32F205/215xx, and STM32F207/217xx devices**

Time	Description	Min	Unit
a	Reset temporization	0.5	ms
b	HSI oscillator startup time	0.0022	ms
c	Bootloader firmware operations	0.01	ms
d	PLL Lock time	0.075	ms
e	Bootloader firmware operations	84	ms
A	Time = a + b + c + d + e	84.5872	ms
B	Connection establishment	54	ms

For the STM32F205xx, STM32F215xx, STM32F207xx and STM32F217xx devices bootloader, the timing values are independent from the HSE crystal frequency. The detection of the HSE crystal frequency value is performed through period measurement using TIM11 timer and HSI internal oscillator.

# 11 Revision history

**Table 27. Document revision history**

Date	Revision	Changes
22-Oct-2007	1	Initial release.
22-Jan-2008	2	<p>All STM32 in production (rev. B and rev. Z) include the bootloader described in this application note.</p> <p>Modified: <a href="#">Section 3.1: Bootloader activation</a> and <a href="#">Section 1.4: Bootloader code sequence</a>.</p> <p>Added: <a href="#">Section 1.3: Hardware requirements</a>, <a href="#">Section 1.5: Choosing the USART baud rate</a>, <a href="#">Section 1.6: Using the bootloader</a> and <a href="#">Section 3.2: Exiting System memory boot mode</a>.</p> <p><a href="#">Note 2</a> linked to Get, Get Version &amp; Read Protection Status and Get ID commands in <a href="#">Table 3: Bootloader commands</a>, <a href="#">Note 3</a> added.</p> <p>Notion of “permanent” (Permanent Write Unprotect/Readout Protect/Unprotect) removed from document. Small text changes.</p> <p>Bootloader version upgraded to 2.0.</p>
26-May-2008	3	<p>Small text changes. RAM and System memory added to <a href="#">Table 3: STM32F10xxx configuration in System memory boot mode</a>.</p> <p><a href="#">Section 1.6: Using the bootloader on page 8</a> removed.</p> <p>Erase modified, <a href="#">Note 3</a> modified and <a href="#">Note 1</a> added in <a href="#">Table 3: Bootloader commands on page 9</a>.</p> <p><a href="#">Byte 3: on page 11</a> modified.</p> <p><a href="#">Byte 2: on page 13</a> modified.</p> <p><a href="#">Byte 2:, Bytes 3-4: and Byte 5: on page 15</a> modified, <a href="#">Note 3</a> modified.</p> <p><a href="#">Byte 8: on page 18</a> modified.</p> <p>Notes added to <a href="#">Section 2.5: Go command on page 18</a>.</p> <p><a href="#">Figure 11: Go command: device side on page 20</a> modified.</p> <p>Note added in <a href="#">Section 2.6: Write Memory command on page 21</a>.</p> <p><a href="#">Byte 8: on page 24</a> modified.</p> <p><a href="#">Figure 14: Erase Memory command: host side</a> and <a href="#">Figure 15: Erase Memory command: device side</a> modified.</p> <p><a href="#">Byte 3: on page 26</a> modified.</p> <p><a href="#">Table 3: Bootloader commands on page 9</a>.</p> <p>Note modified and note added in <a href="#">Section 2.8: Write Protect command on page 27</a>.</p> <p><a href="#">Figure 16: Write Protect command: host side</a>, <a href="#">Figure 17: Write Protect command: device side</a>, <a href="#">Figure 19: Write Unprotect command: device side</a>, <a href="#">Figure 21: Readout Protect command: device side</a> and <a href="#">Figure 23: Readout Unprotect command: device side</a> modified.</p>
29-Jan-2009	4	<p>This application note also applies to the STM32F102xx microcontrollers.</p> <p>Bootloader version updated to V2.2 (see <a href="#">Table 4: Bootloader versions</a>).</p>

Table 27. Document revision history (continued)

Date	Revision	Changes
19-Nov-2009	5	<p>IWDG added to <a href="#">Table 3: STM32F10xxx configuration in System memory boot mode</a>. <a href="#">Note</a> added.</p> <p>BL changed bootloader in the entire document.</p> <p>Go command description modified in <a href="#">Table 3: STM32F10xxx configuration in System memory boot mode</a>.</p> <p>Number of bytes awaited by the bootloader corrected in <a href="#">Section 2.4: Read Memory command</a>.</p> <p>Note modified below <a href="#">Figure 10: Go command: host side</a>.</p> <p>Note removed in <a href="#">Section 2.5: Go command</a> and note added.</p> <p>Start RAM address specified and note added in <a href="#">Section 2.6: Write Memory command</a>. All options are erased when a Write Memory command is issued to the Option byte area.</p> <p><a href="#">Figure 11: Go command: device side</a> modified.</p> <p><a href="#">Figure 13: Write Memory command: device side</a> modified.</p> <p>Note added and bytes 3 and 4 sent by the host modified in <a href="#">Section 2.7: Erase Memory command</a>.</p> <p>Note added to <a href="#">Section 2.8: Write Protect command</a>.</p>
09-Mar-2010	6	<p>Application note restructured. Value line and connectivity line device bootloader added (Replaces AN2662).</p> <p><a href="#">Introduction</a> changed. <a href="#">Glossary</a> added.</p>
20-Apr-2010	7	<p><a href="#">Related documents</a>: added XL-density line datasheets and programming manual.</p> <p><a href="#">Glossary</a>: added XL-density line devices.</p> <p><a href="#">Table 2</a>: added information for XL-density line devices.</p> <p><a href="#">Section 4.1: Bootloader configuration</a>: updated first sentence.</p> <p><a href="#">Section 5.1: Bootloader configuration</a>: updated first sentence.</p> <p>Added <a href="#">Section 6: STM32F101xx and STM32F103xx XL-density device bootloader</a>.</p> <p><a href="#">Table 18</a>: added information for XL-density line devices.</p>
08-Oct-2010	8	Added information for high-density value line devices in <a href="#">Table 2</a> and <a href="#">Table 18</a> .
14-Oct-2010	9	Removed references to obsolete devices.
26-Nov-2010	10	Added information on ultralow power devices.
13-Apr-2011	11	<p>Added information related to STM32F205/215xx and STM32F207/217xx devices.</p> <p>Added <a href="#">Section 10: Bootloader timing characteristics</a></p>
06-Jun-2011	12	<p>Updated:</p> <ul style="list-style-type: none"> <li>– <a href="#">Table 11: STM32L15xxx bootloader versions</a></li> <li>– <a href="#">Table 12: STM32F2xx configuration in System memory boot mode</a></li> <li>– <a href="#">Table 14: STM32F2xx bootloader V2.x versions</a></li> <li>– <a href="#">Table 17: STM32F2xx bootloader V3.x versions</a></li> </ul>

**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.

© 2011 STMicroelectronics - All rights reserved

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](http://www.st.com)

