AN4852
Application note

Programming an external Flash memory using the UART bootloader built-in STM32 microcontrollers

Introduction

This application note explains how to program an external Quad-SPI Flash memory using the internal bootloader via the UART protocol.

A user boot-code that makes possible the programming of an external Quad-SPI memory has been developed and downloaded in the embedded SRAM to keep the Flash memory ready for other tasks.

The X-CUBE-EXTBOOT firmware, available on *www.st.com*, implements the same protocol of the internal UART bootloader.

The following documents, available on *www.st.com*, are considered as reference:

- AN3155, “USART protocol used in the STM32 bootloader”
- AN2606, “STM32 microcontroller system memory boot mode”

The STM32446E-EVAL boards have been used to develop and validate the firmware.

Contents

List of tables

List of figures

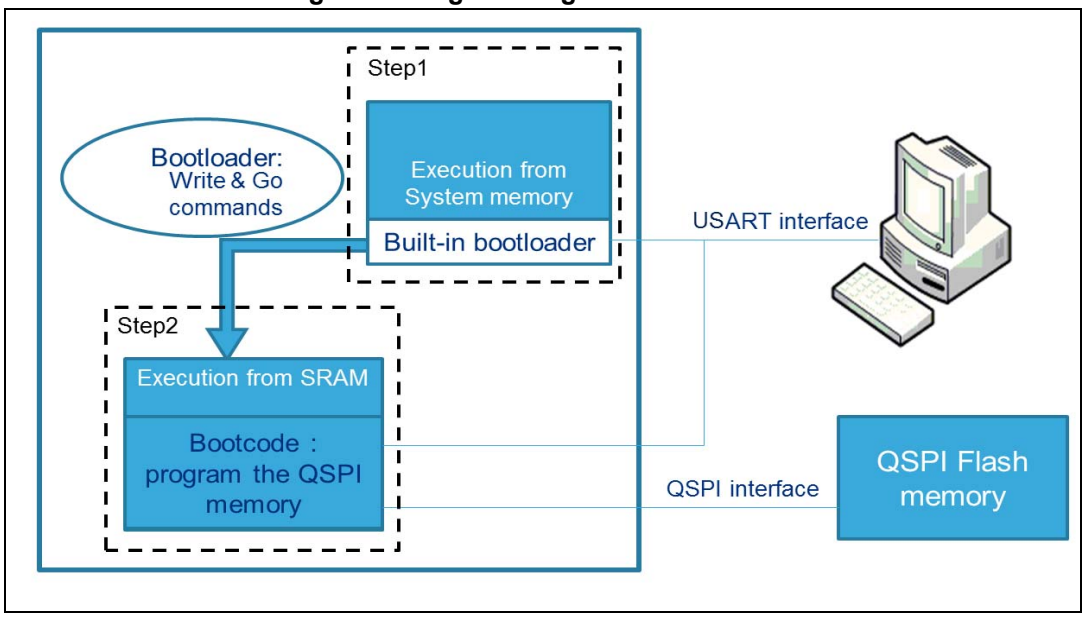
1 Implementation

The purpose of this firmware is to program new firmware/data in an external Flash memory, using the same protocol of the internal bootloader:

Step1: Via the USART interface and the “Write” command, the ST internal bootloader downloads the developed binary code into the internal RAM memory, and then, using the “Go” command, it jumps to the entry point of this binary code to execute it (see *Figure 1*).

Step2: The new bootcode runs from RAM and enables to program the external Quad-SPI Flash memory.

Figure 1. Programming method overview



To reach this goal, user must use the Flash memory loader demonstrator tool, modified to support programming the internal RAM and the Quad-SPI Flash memory.

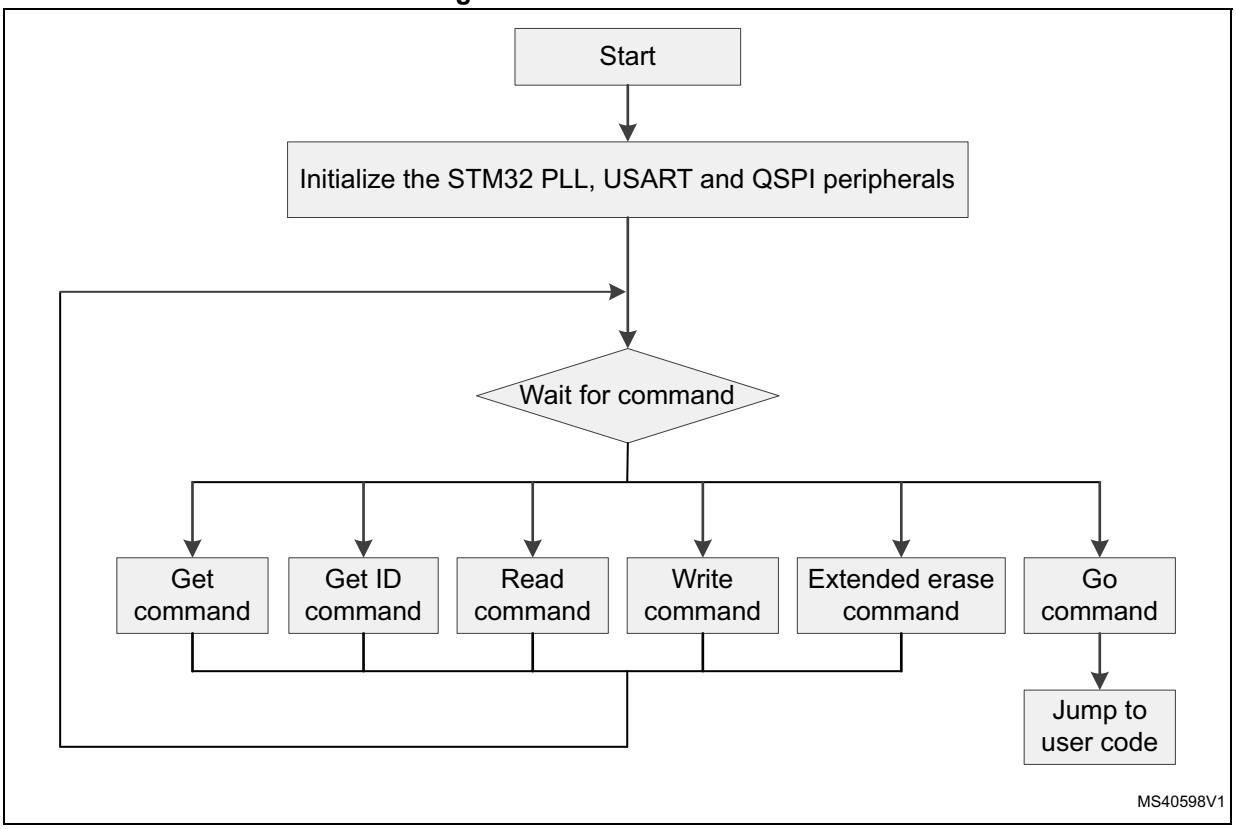
1.1 Overview

In this application note the user bootcode is downloaded in the internal RAM using the internal bootloader, so that the Flash memory content is not altered.

This bootcode, located in the internal RAM, implements commands allowing the user to program the Quad-SPI Flash memory.

Figure 2 describes the user bootcode firmware and lists the different implemented commands.

Figure 2. Firmware flow-chart



1.2 Supported commands

The commands supported by the developed firmware are listed in *Table 1*. Each of them will be described in detail in this section.

Table 1. Supported commands

Command	Code	Description
Get	0x00	Gets the bootloader version and the commands supported by this version
Get ID	0x02	Gets the chip ID
Go	0x21	Jumps to the loaded application code located in an address specified by the user
Read	0x11	Reads up to 256 bytes of the external Quad-SPI memory starting from an address specified by the user
Write	0x31	Writes up to 256 bytes in the external Quad-SPI memory starting by an address chosen by the user
Extended Erase	0x44	Erases (from one to all) sectors of an external Quad-SPI memory

Communication safety

All communication from the programming tool to the device is verified in the following ways:

- Checksum: all received bytes are XOR-ed. A byte containing the computed XOR of all previous bytes is added at the end of each communication (checksum byte).
- For each command, the host sends a byte and its complement.
- UART: a parity check is active (even parity).

Each packet is either accepted (ACK answer) or discarded (NACK answer).

- ACK = 0x79
- NACK = 0x1F

1.2.1 Get command

The Get command allows the user to get the version of the serial peripheral communication protocol used by the bootloader and the supported commands.

When the STM32 microcontroller receives the code of Get command and its corresponding checksum (0x00 – 0xFF), it reacts as follows:

- Transmits an acknowledgment to the host
- Sends the number of bytes to be sent -1: 0x06 (six commands are supported by the new firmware)
- Sends the used bootloader version
- Lists the supported commands codes (indicated in *Table 1*) to the host.

1.2.2 Get ID command

This command allows the user to get the Chip identification. When the microcontroller receives this command, it transmits the Product ID to the host.

The STM32 sends the bytes as follows:

- ACK byte: After a correct reception of the command code and its convenient checksum (0x02 – 0xFD).
- The number of bytes to send -1: 0x01
- The product ID (2bytes: 0x04 – 0x21, as STM32F446 has been used in this case)
- ACK byte

1.2.3 Go command

The Go command is used to jump to a specified address in the Quad-SPI external memory, and to execute the code downloaded there.

When the STM32 receives the Go command and its checksum correctly (0x21 – 0xDE):

- It verifies if the user area in the Flash memory is read protected. If this is the case, it sends a NACK and aborts the operation. Otherwise (Flash memory is not read protected), it transmits an acknowledgment.
- Then, the MCU waits for the host to send the address where the code will be loaded (coded on 4 bytes) and a checksum byte.
- The STM32 verifies the validity of the received address and whether the checksum is exact: if those conditions are approved, an ACK byte is sent to the host and a software remapping is performed. The CPU program counter jumps automatically to the

indicated address and executes the code loaded there; else it aborts the command after a NACK byte has been sent to the host.

1.2.4 Read command

The read command is used to read data from any valid memory address of the external Quad-SPI memory.

When the STM32 receives the Read Memory command, it verifies if the user area in the internal Flash memory is read protected or not. If it is protected, the STM32 sends a NACK byte and aborts the command. Otherwise, it transmits an ACK byte to the host, then waits for the start address (coded on 4 bytes) and its checksum byte, checks the validity of the received address and whether the checksum is correct or not. If the verification is successful, the STM32 transmits the needed data to the host; else it sends a NACK before exiting from the command.

This function can read up to 256 bytes from the memory starting from the given address.

The read operation is detailed in the flow-chart shown in *Figure 3*.

Additional configurations needed before reading from the Quad-SPI memory

The reading sequence is carried out using “QUAD I/O fast read” command. Accordingly, when the STM32 receives the start address and before reading data from the Quad-SPI memory, volatile configuration register must be configured with new dummy cycles value (Dummy clock cycles = 10, following QSPI memory protocol requirements).

1.2.5 Write command

The Write Memory command is used to write data to any valid memory address in the external Quad-SPI memory. When the STM32 receives this command, and if the user area in the internal Flash memory is not read protected, it transmits an ACK byte to the host and carries out the write operation; otherwise, it sends a NACK byte and aborts the command.

This function can write up to 256 bytes starting from the given address.

The write process is detailed in the flow-chart shown in *Figure 4*.

Additional configurations needed before programming the Quad-SPI memory

When the STM32 receives the start address and before writing into Quad-SPI memory, some configurations need to be fixed:

- A "Write Enable" command must be executed to deactivate the memory protection before sending the write command; otherwise the Quad-SPI memory will not accept any memory programming
- Writing sequence is done using the command Extended Quad input fast program
- Wait for end of program should be implemented

Figure 3. Read command, STM32 side

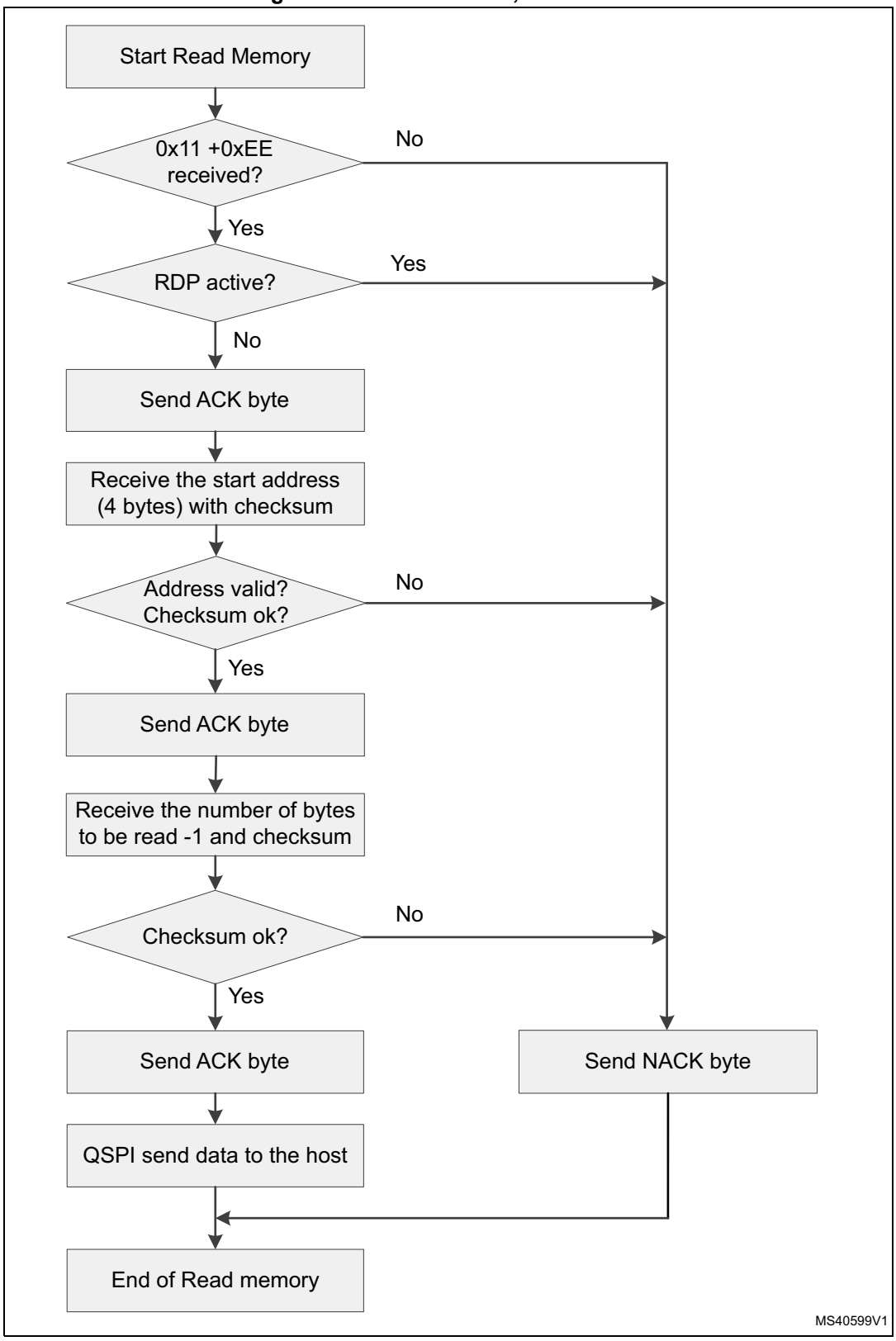
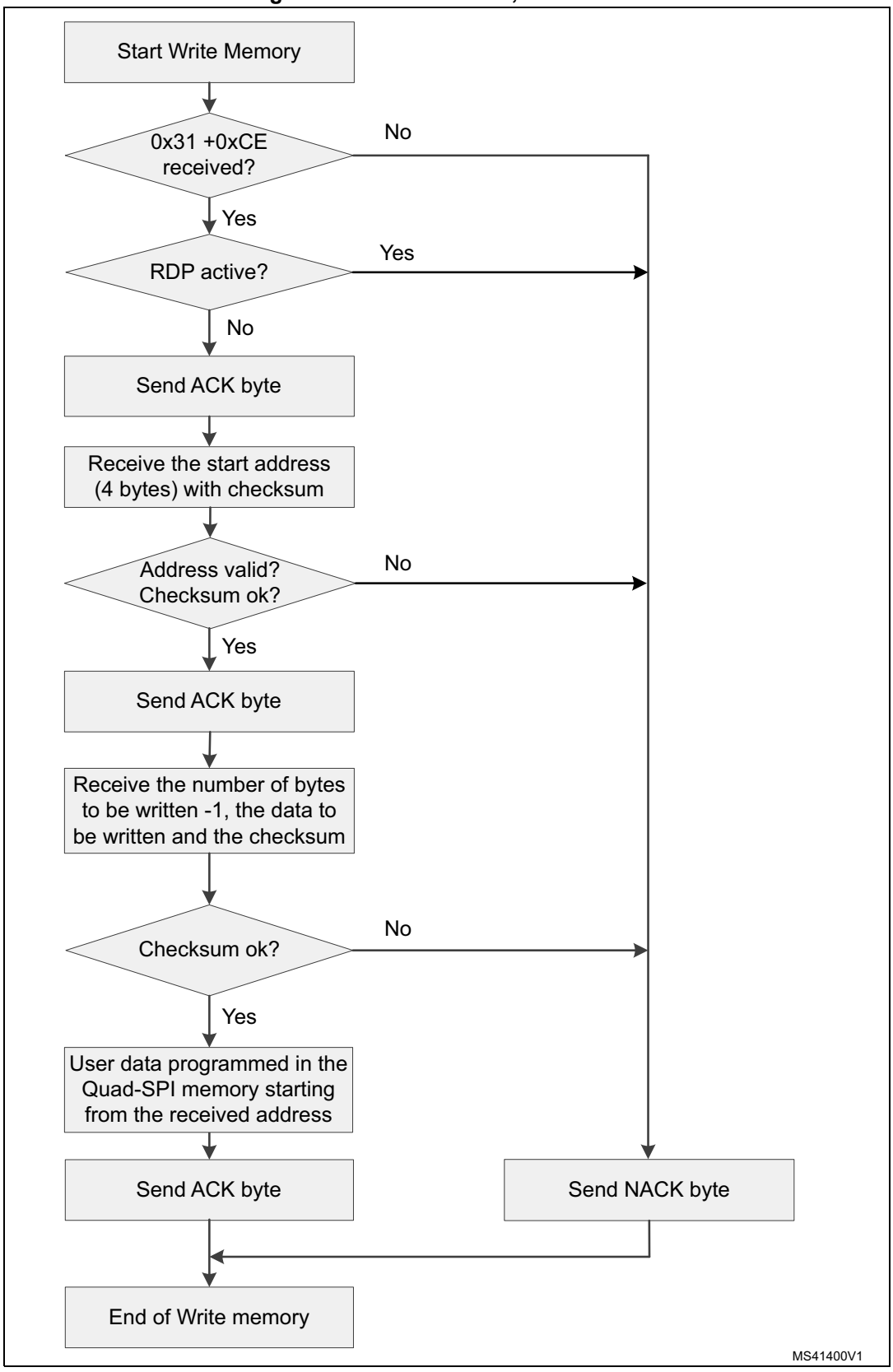


Figure 4. Write command, STM32 side



1.2.6 Extended erase command

The Extended Erase Memory command allows the host to erase the external memory. This operation is performed using two bytes addressing mode and the erase of the Quad-SPI memory is executed sector by sector.

When the STM32 receives correctly this command and its checksum (0x44 - 0xBB), it transmits an acknowledgment to the host and operates as follows:

- Waits for the number of sectors to be erased diminished by 1 (coded on two bytes): N sectors to erase
- For the N sectors, the STM32 receives each sector number (coded on two bytes) and a checksum byte (XOR of all the received bytes)
- If the calculated checksum is equal to the received one, the STM32 erases the selected sectors and sends an acknowledgment byte to the host. Otherwise, it sends a NACK byte and quits the operation.

Additional configurations needed before erasing Quad-SPI sectors

To erase sectors from the Quad-SPI memory, some configurations need to be set:

- “Write enable” command must be called, otherwise the Quad-SPI memory ignores the erase command, and no error bits are sent to indicate an operation failure.
- Erasing operation is processed using the sector erase command.
- Wait for end of program should be implemented.

2 How to use the user boot code?

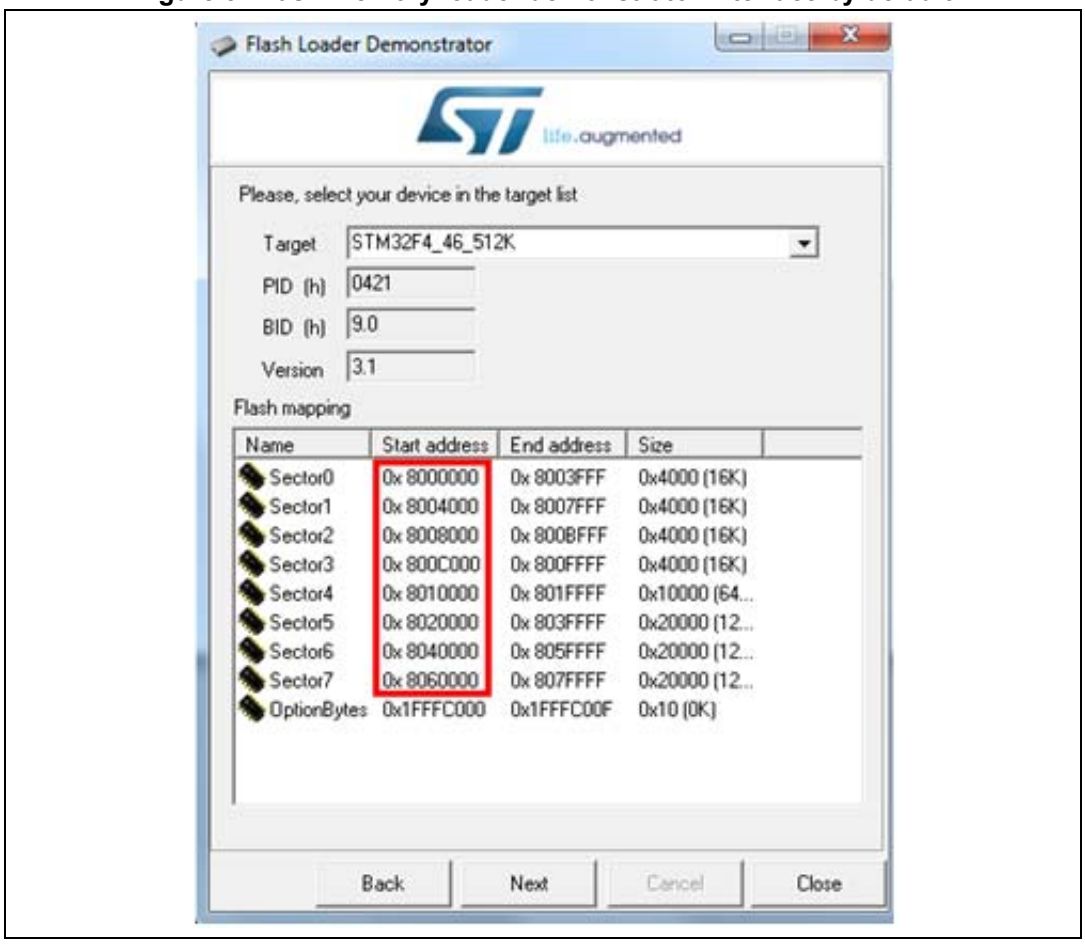
This section describes the steps needed to execute the commands listed in *Table 1*.

2.1 Adding new Flash loader demonstrator mapping description files

The Flash memory loader demonstrator tool will be used to download the bootcode into RAM memory, and then to specify the memory area to be read/erased, or where the data will be written.

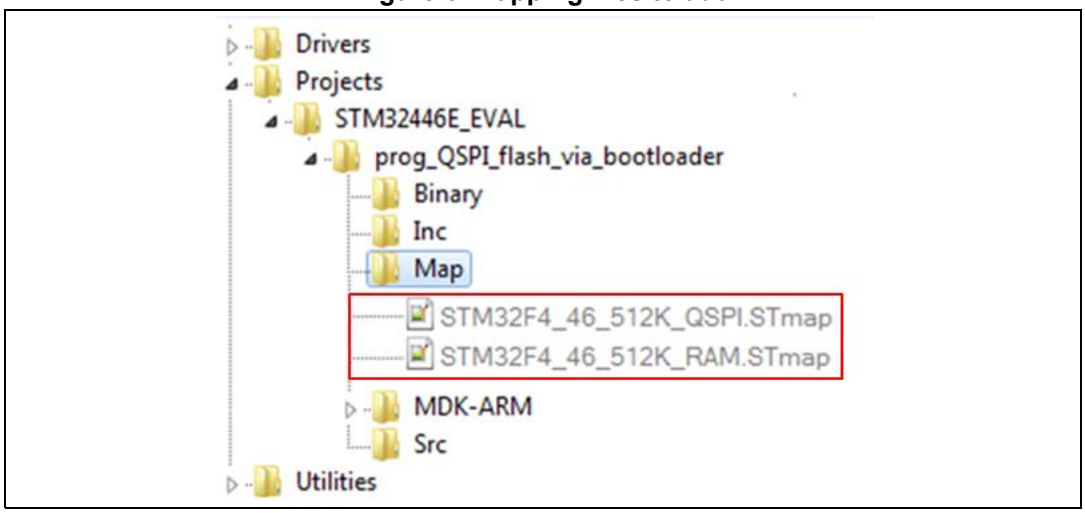
This tool is configured by default for reprogramming the internal Flash memory, the mapping description file contains only sections of the internal memory (see *Figure 5*).

Figure 5. Flash memory loader demonstrator interface by default



Therefore, two other mapping description files are added, one for the RAM and the other for programming the Quad-SPI memory. These files are located in the X-CUBE-EXTBOOT firmware package under the path shown in *Figure 6*. Copy these files in the Map folder, which contains the mapping description files of all the supported devices. This folder is located in the Flash loader demonstrator tool installation directory.

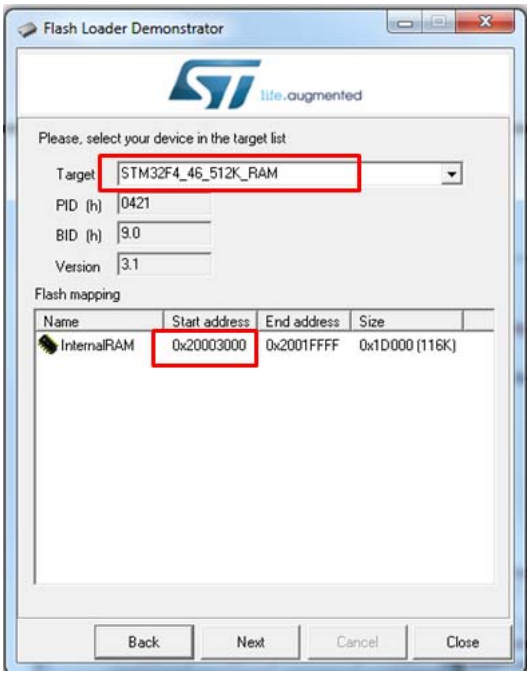
Figure 6. Mapping files to add



Adding a RAM area

The first mapping description file enables the tool to program the RAM area where the bootcode will be downloaded (*Figure 7*).

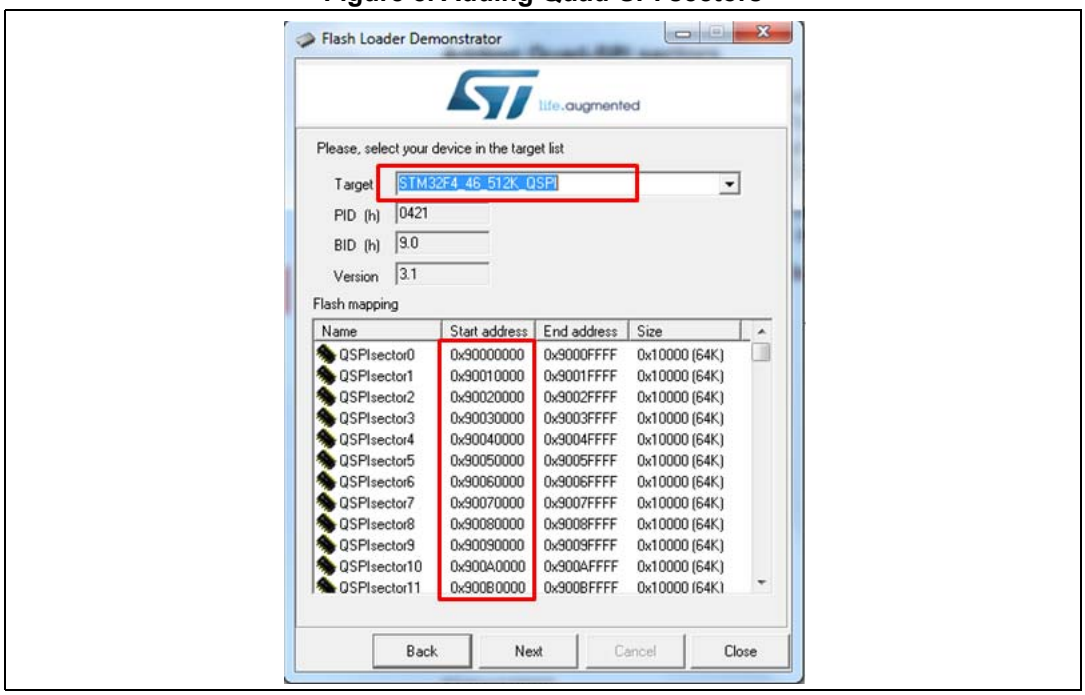
Figure 7. Adding RAM area



Adding Quad-SPI sectors

The second mapping description file enables the loader demonstrator to program the Quad-SPI memory (*Figure 8*).

Figure 8. Adding Quad-SPI sectors

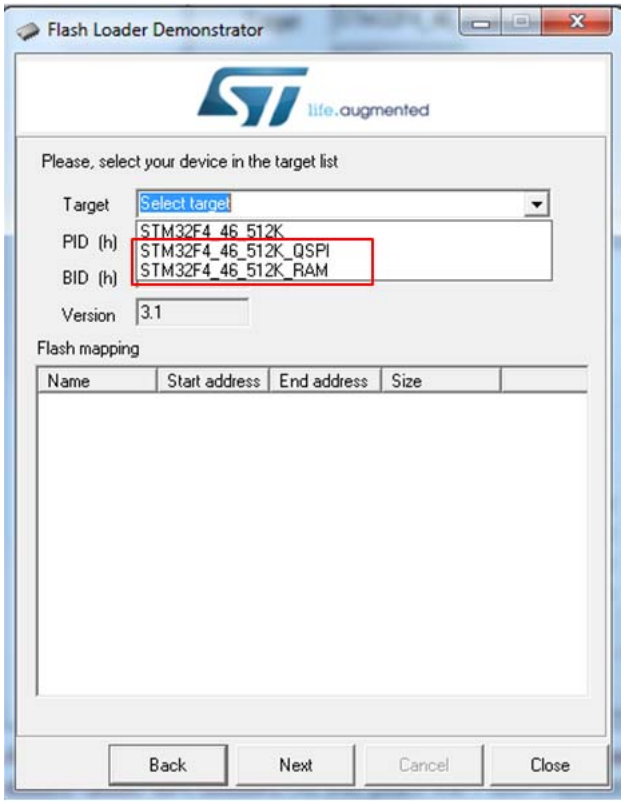


The STM32F4 Quad-SPI interface is able to manage up to 256 Mbit of Flash memory, starting from 0x9000 0000 up to 0X9FFF FFFF. The external Quad-SPI Flash memory capacity can be up to 4 GBytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 MBytes.

In this project, the used external memory is a Quad-SPI memory from Micron, having a density of 32 MBytes, configured as 512 sectors (64 KBytes each). The developed bootcode allows the user to program only the first 16 MBytes (the first 256 sectors) of the Quad-SPI memory.

At this point the Flash memory loader demonstrator tool is able to download the code into internal RAM and to program the Quad-SPI memory. For each operation, the appropriate target device selection should be set (*Figure 9*).

Figure 9. Device selection



2.2 Loading the user boot code into RAM memory

Programming is performed via the free ST Flash memory loader demonstrator tool, after having added the two mapping descriptor files to the installation directory, as shown in *Section 2.1*. This tool will be used to download the developed “user bootcode” into internal RAM and ensure the jump to execute it.

To load the user boot code into RAM, follow the procedure below:

- The Flash memory loader primarily configures the serial communication (*Figure 10*).
- Target device selection must be specified (*Figure 7*).
- Download image to internal RAM: choose the binary of the code to be loaded and select the convenient RAM address and do not forget to tick the “jump to user program” box (*Figure 11*).
- If the procedure is correctly followed, the user boot code is written into the chosen RAM address, and the window shown in *Figure 12* appears.
- Close the previous window. Now, the CPU has already jumped to the indicated RAM address and begun the execution of the code existing there. LED 1 is ON to indicate the beginning of the bootcode execution.

Figure 10. Communication port selection and set-up

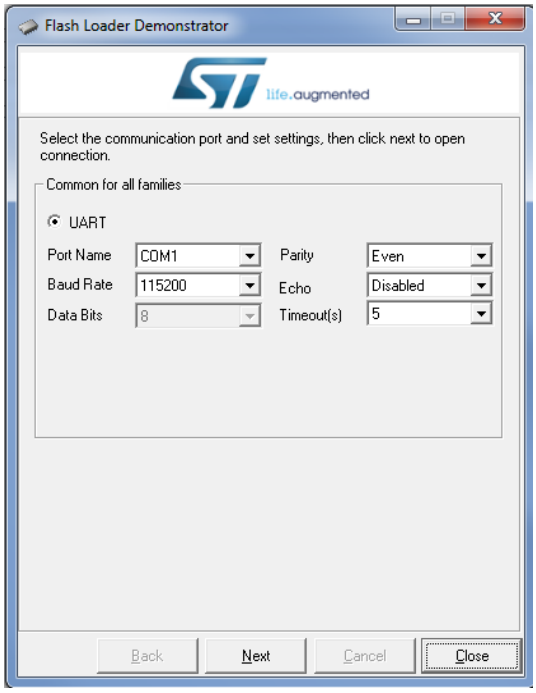


Figure 11. Download image

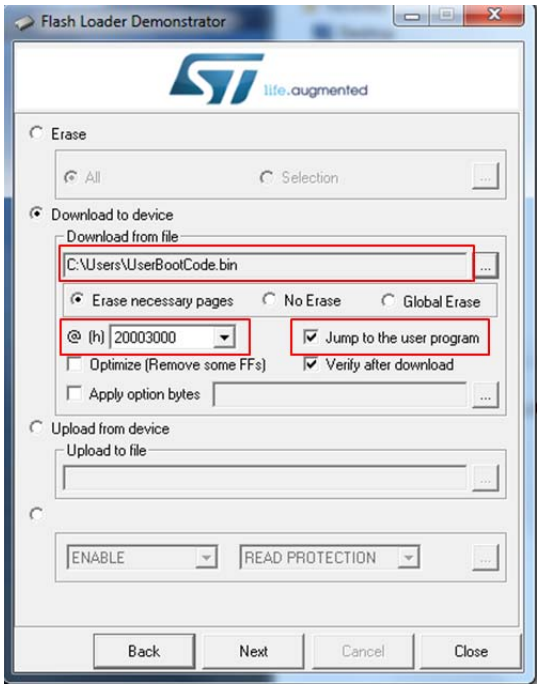
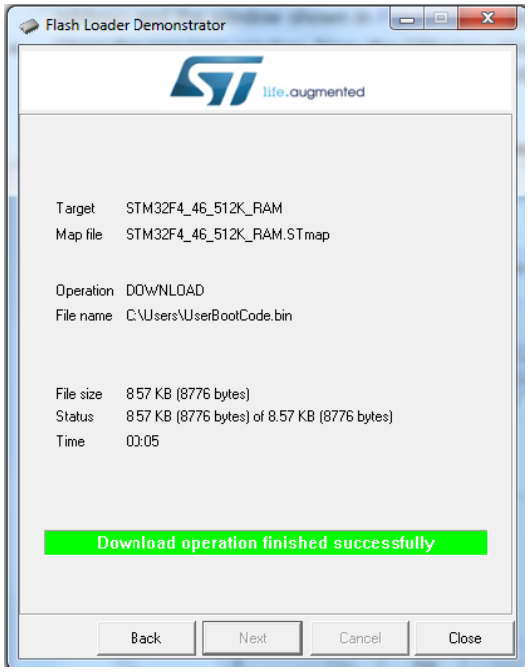


Figure 12. Successful download operation



2.3 Running user boot code using the Flash memory loader

At this stage, the CPU is waiting for the host to send the command codes to perform Read, Write or Erase operations to/from the Quad-SPI memory.

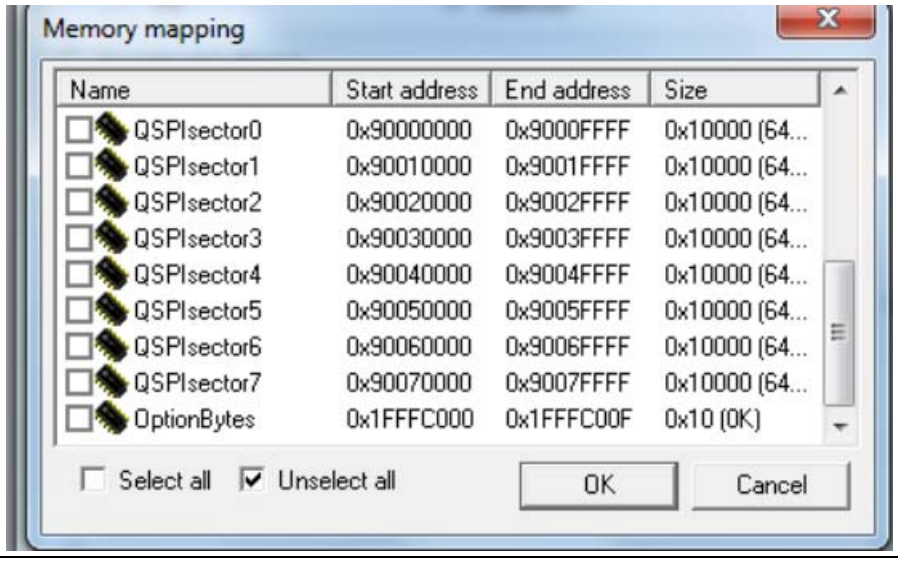
Reopen the Flash memory loader demonstrator tool, choose the target that enables it to program the Quad-SPI memory (*Figure 8*), and then tick the command to operate.

2.3.1 Erase operation

Click on “Erase” box (*Figure 13*), then select the memory area to erase (*Figure 14*).

Figure 13. Erase operation

Figure 14. Tick sectors to erase

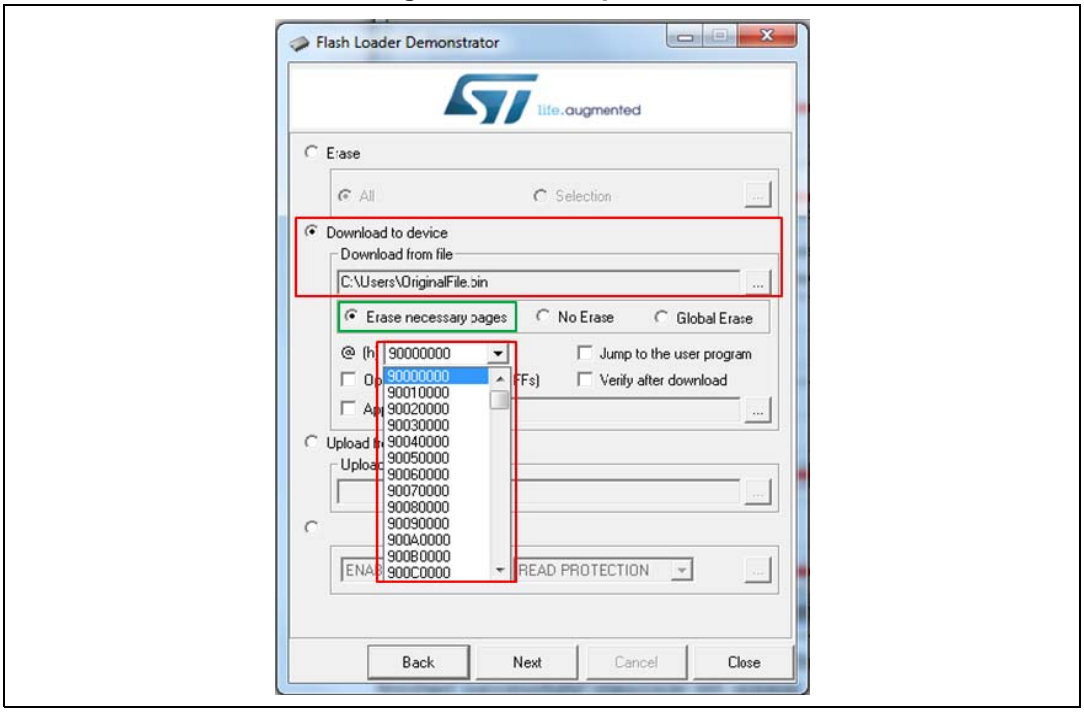


Click on “OK”, then “Next”. The message “Erase operation finished successfully” appears and the contents of the selected sectors are erased.

2.3.2 Write operation

Click on “Download to device” box, choose the file that will be written, and point out the address where it will be written (*Figure 15*). Do not forget to tick the “Erase necessary pages” box, as this will erase the sectors where the file will be written. Then click on “Next”.

Figure 15. Write operation



The contents of the file are copied into the chosen address and a “Download operation finished successfully” message appears. Info about the file size and the duration of download are provided as well.

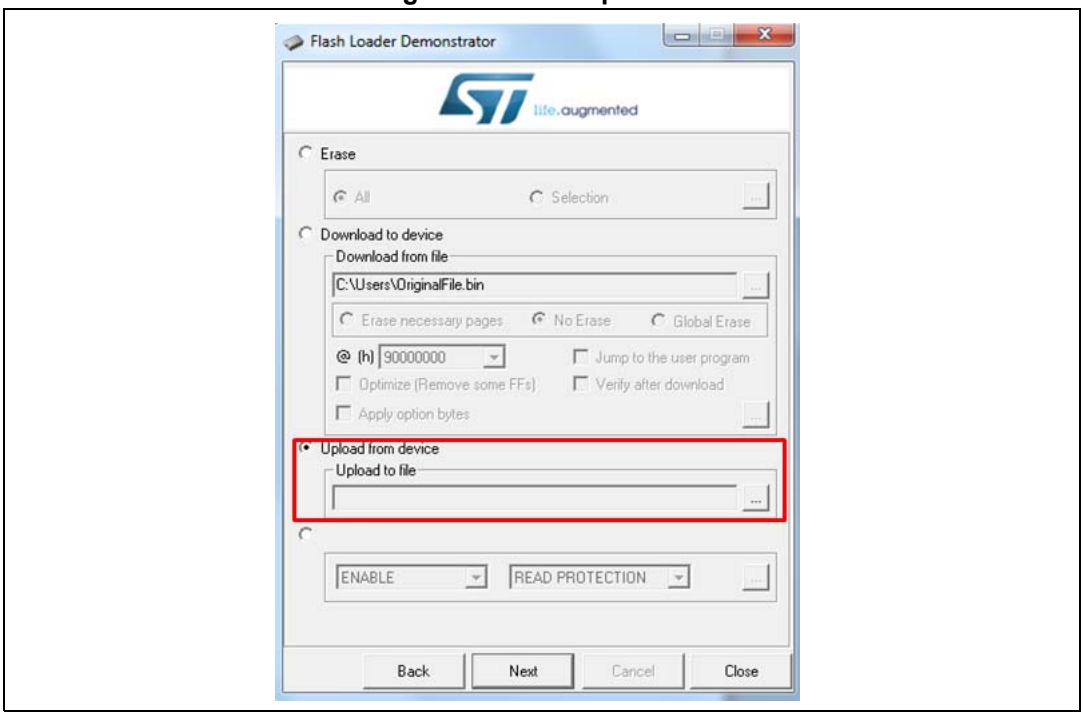
Note: To be sure that the file has been correctly written, the user can tick the “Verify after download” case before launching the write operation (this will launch the verification process).

Note: If the user wants to execute the programmed code just after the Write operation has completed, the “Jump to the user program” case must be ticked before launching the Write operation.

2.3.3 Read operation

Click on “Upload from device” box (*Figure 16*), then click on “Upload to file” to choose the memory areas to be read and to create the file where the contents of those memory areas will be saved. Click on “OK”, then on “Next”.

Figure 16. Read operation



The contents of the chosen memory areas are saved in the created file and an “Upload operation finished successfully” message appears. Additional information about the file size and the duration of upload are provided as well.

3 Conclusion

With the built-in UART bootloader available in STM32 microcontrollers, users program the RAM and the internal Flash memory.

Thanks to the firmware described in this application note, users have the additional possibility of programming an external Quad-SPI Flash memory and to develop their own bootloader to program any external memory connected to communication peripherals such as I2C, SPI, FSMC/FMC, Quad-SPI.

4 Revision history

Table 2. Revision history

Date	Revision	Description of changes
17-Jun-2016	1	Initial release.