



---

## STM32CubeProgrammer



A quality version of this page, approved on *30 March 2021*, was based off this revision.

This article gives a short introduction to the official STM32CubeProgrammer tool: What are the flashing principles? Where to download the software? How to install it? How to use it?




# STM32 CubeProgrammer

## Contents

1 STM32CubeProgrammer overview .....	3
2 STM32CubeProgrammer installation .....	5
2.1 Installing the STM32CubeProgrammer tool .....	5
2.2 Preparing the USB serial link for flashing .....	6
3 Flash programming principles .....	8
4 How to flash with STM32CubeProgrammer .....	9
5 FAQs .....	10
5.1 Where to find the STM32CubeProgrammer user manual .....	10
5.2 How to check if the DFU driver is functional .....	10
5.3 How to proceed when the DFU driver installation fails on Windows host PC .....	10
5.4 How to find the DEVICE_PORT_LOCATION parameter value for the USB link .....	12
5.5 How to explore all the partitions of a populated microSD card .....	13
5.6 How to update partitions .....	13
5.7 How to populate a microSD card inserted in a <b>Linux host PC</b> .....	13
5.8 How to fuse STM32MP15x OTP .....	13
5.8.1 Connection .....	13
5.8.2 Display status .....	14
5.8.3 Update value .....	17
5.8.4 Update lock .....	18
5.9 How to program STPMIC NVM .....	18



---

## 1 STM32CubeProgrammer overview

---

STM32CubeProgrammer is the official STMicroelectronics tool for creating partitions into any Flash device available on STM32 platforms.

Once created, STM32CubeProgrammer allows populating and updating the partitions with the prebuilt binaries.

The connection between the host PC and the board can be done through UART or USB serial links.

Any Flash device supported on STM32MPU boards can be flashed using STM32CubeProgrammer. For example, for the STM32MP157 Evaluation board:

1. microSD™ card
2. eMMC
3. NAND Flash memory
4. NOR Flash memory

### Information

During development, it is recommended to use a microSD card; you can then switch to eMMC or NAND Flash memory.

More details on supported Flash memory technologies and mapping can be found in Flash mapping articles, e.g. [STM32MP15 Flash mapping](#).





## 2 STM32CubeProgrammer installation

### 2.1 Installing the STM32CubeProgrammer tool

	STM32CubeProgrammer for Linux <sup>®</sup> host PC	STM32CubeProgrammer for Windows <sup>®</sup> host PC
<b>Download</b>	<p><b>Version 2.9.0</b></p> <ul style="list-style-type: none"> <li>• Browse <a href="#">SetupSTM32CubeProgrammer</a> link and follow instructions to download the package, you need a myST account.</li> <li>• Download the archive file on your host PC in a temporary directory</li> <li>• Uncompress the archive file to get the STM32CubeProgrammer installers:</li> </ul> <p><b>PC</b> <code>\$&gt;unzip en.stm32cubeprog.zip</code></p>	
<b>Installation</b>	<ul style="list-style-type: none"> <li>• Execute the Linux installer, which guides you through the installation process.</li> </ul> <pre style="border: 1px dashed gray; padding: 5px;">\$&gt; ./SetupSTM32CubeProgrammer-2.9.0.linux</pre> <ul style="list-style-type: none"> <li>• The path to the STM32CubeProgrammer binary must be added to the PATH environment variable <ul style="list-style-type: none"> <li>• either in each Terminal program in which the STM32CubeProgrammer binary needs to be used, using the following command:</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Execute the Windows installer, which guides you through the installation process.</li> </ul>



	STM32CubeProgrammer for Linux® host PC	STM32CubeProgrammer for Windows® host PC
	<pre>\$&gt; export PATH=&lt;my STM32CubeProgrammer install directory&gt; /bin:\$PATH</pre> <ul style="list-style-type: none"> <li>• or once for all by creating a link to the STM32CubeProgrammer binary in a directory already present in PATH. For example, if "/home/bin" is in the PATH environment variable, run the following command:</li> </ul> <pre>\$&gt; ln -s &lt;my STM32CubeProgrammer install directory&gt; /bin /STM32_Programmer_CLI /home/bin /STM32_Programmer_CLI</pre>	
<b>User manual</b>	<ul style="list-style-type: none"> <li>• Instructions to follow for using the STM32CubeProgrammer can be found in user manual, UM2237 available from ST web site, or in <a href="#">STM32CubeProgrammer#How to flash with STM32CubeProgrammer</a> article.</li> </ul>	
<b>Detailed release note</b>	<ul style="list-style-type: none"> <li>• Details about the content of this tool version are available from ST web site at <a href="#">Release Note</a> .</li> </ul>	

## 2.2 Preparing the USB serial link for flashing

It is recommended to use the USB (in DFU mode) for flashing rather than the UART, which is too slow.

Below indications on how to install the USB in DFU mode under Linux and Windows OS, respectively.

- **For Linux host PC** or Windows host PC with VMWare:

The libusb1.0 package (including USB DFU mode) must be installed to be able to connect to the board via the USB port. This is achieved by typing the following command from the host PC terminal:

```
PC $> sudo apt-get install libusb-1.0-0
```



---

To allow STM32CubeProgrammer to access the USB port through low-level commands, proceed as follows:

```
PC $> cd <your STM32CubeProgrammer install directory>/Drivers/rules
PC $> sudo cp *.* /etc/udev/rules.d/
```

- **For Windows host PC:**

Run the “STM32 Bootloader.bat” file to install the STM32CubeProgrammer DFU driver and activate the STM32 microprocessor device in USB DFU mode. This driver (installed by STM32 Bootloader.bat) is provided within the STM32CubeProgrammer release package. It is located in the DFU driver folder, \Drivers\DFU\_Driver.

In case of issue, refer to [How to proceed when the DFU driver installation fails on Windows host PC](#).

To validate the installation, the DFU driver functionality can be verified by following the FAQ instructions provided in [how to check if the DFU driver is functional](#).

**STM32CubeProgrammer is now ready to use: to know more about this tool, please continue reading; otherwise, jump to the [Starter or Distribution Package](#) article corresponding to your board: [Category:Starter Package](#) or [Category:Distribution Package](#).**



---

### 3 Flash programming principles

---

Flash programming consists in transferring the binaries stored on the host computer into the platform Flash memory(ies), via a serial interface.

This operation requires a communication interface between the **STM32CubeProgrammer** and an **embedded programming service**.

**The selection of the binaries to download and the Flash memory destination is done through the flashlayout.tsv file.**

The STM32CubeProgrammer tool uses the Flashlayout.tsv file as an input.

The Flashlayout includes a formal description of the partitions (ID, naming, type, offset) as well as the identification of the Flash memory to be populated.

Some default "tsv" files aligned with the STM32 Flash memory mapping (e.g. STM32MP15\_Flash\_mapping) are provided in the STM32CubeProgrammer tool. They can be used as a starting point and can be further adapted to specific application needs.

More examples as well as information on the description format are available in [STM32CubeProgrammer flashlayout](#).

See AN5275: USB DFU/USART protocols used in STM32MP1 Series bootloaders for protocol details.





## 4 How to flash with STM32CubeProgrammer

Once the STM32CubeProgrammer has been installed, complete flashing procedures for STM32 boards are available in Starter Package and Distribution Package articles. Select the article corresponding to your STMicroelectronics platform.

The generic syntax of the STM32CubeProgrammer command is explained below.

- With Linux host PC, the main command is:

```
PC $> ./bin/STM32_Programmer_CLI + options given below
```

- With Windows host PC, the main command is:

```
PC $> ./bin/STM32_Programmer_CLI.exe + options given below
```

The Linux related information provided below can be extrapolated for other host PC operating systems:

- Complete flashing command (for Linux host PC):

```
PC $> STM32_Programmer_CLI -c port=<DEVICE_PORT_LOCATION> -w [<file.tsv>]
where
w:
  Write
  <file.tsv>:PathToFlashlayout/flashlayout.tsv file (see above)
             if Flashlayout and binaries are not in the same directory, then the path to the
Flashlayout files must be precised.
  <DEVICE_PORT_LOCATION>:
             e.g. usb1 (case sensitive): for other options, please refer to How to find the
DEVICE\_PORT\_LOCATION parameter value for the USB link.
```

Populating all partitions can take a few minutes (depending mainly on the rootfs size).



## 5 FAQs

### 5.1 Where to find the STM32CubeProgrammer user manual

The user manual is available in ST deliveries under <STM32CubeProgrammer installation directory>/doc/UM2336.pdf (see Installing the STM32CubeProgrammer tool).

### 5.2 How to check if the DFU driver is functional

**Prerequisites:** STM32CubeProgrammer installation completed and STM32 board connected to the host PC.

First, activate the USB driver on Ubuntu on VMware Workstation Player (use VM Player menu "removable devices" to connect to STMicroelectronics DFU in HS mode), then execute the following command and check the result:

```
PC $> STM32_Programmer_CLI -l usb
```

- If the DFU is functional, the result is similar to the following

```
-----
STM32CubeProgrammer <tool version>
-----

Total number of available STM32 device in DFU mode: 1

Device Index      : USB1
USB Bus Number    : 002
USB Address Number : 008
Product ID        : USB download gadget@Device ID /0x500, @Revision ID /0x0000
Serial number     : 0000000000
Firmware version  : 0x0110
Device ID         : 0x0500
```

- Otherwise, the DFU is not functional and the result is the following:

```
-----
STM32CubeProgrammer <tool version>
-----

Warning: No STM32 device in DFU mode connected
```

### 5.3 How to proceed when the DFU driver installation fails on Windows host PC

**Prerequisites:** STM32CubeProgrammer installation completed.

The STM32 Bootloader.bat allows to install the DFU driver on a Windows PC (see Preparing the USB serial link for flashing). In case of failure, here is a typical error log:

```

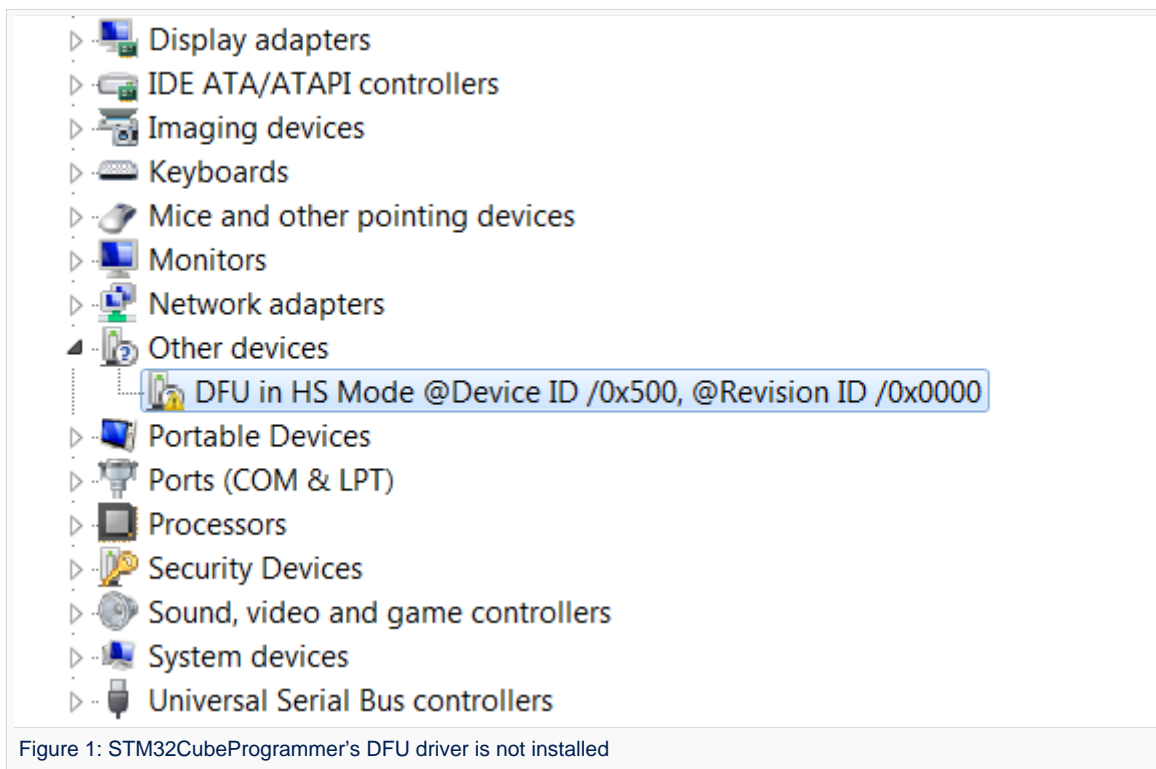
c:\demo\MPU\stm32_programmer_package_v1.0.3_MPU\Drivers\DFU_Driver>"STM32 Bootloader.bat"
c:\demo\MPU\stm32_programmer_package_v1.0.3_MPU\Drivers\DFU_Driver>echo off Microsoft PnP
Utility
Processing inf :          DFU_in_HS_Mode.inf
Failed to install the driver on any of the devices on the system : No more data is
available.
Total attempted:          1
Number successfully imported: 0

```

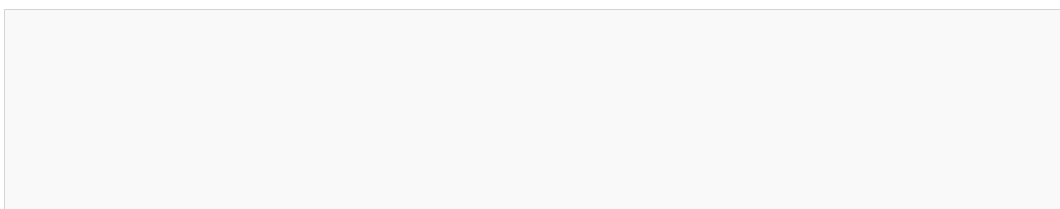
This error occurs either because the STM32CubeProgrammer DFU driver is already installed or because the DFUSE driver is installed.

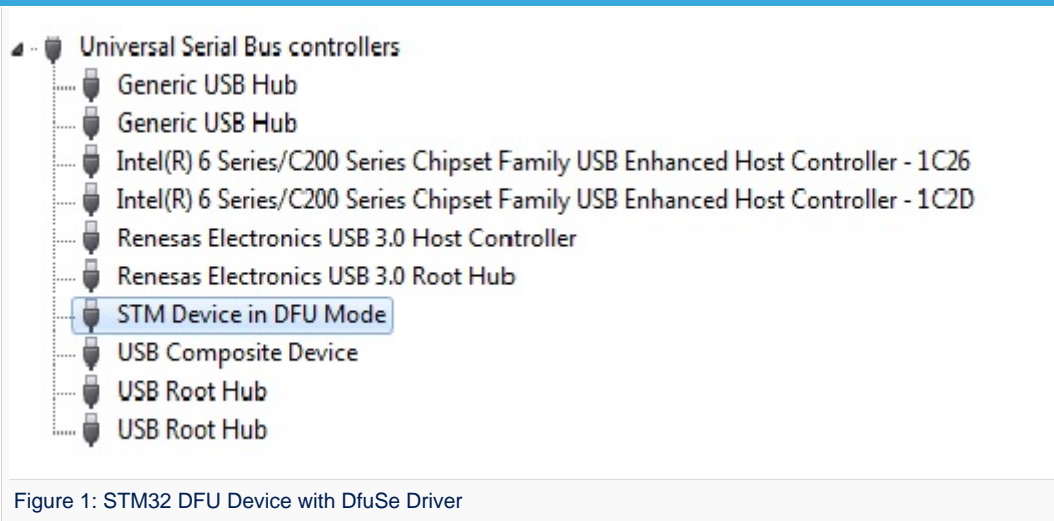
To verify if the STM32CubeProgrammer DFU driver is correctly installed on your Windows host PC, please proceed as follow:

1. Connect the board to the Windows host PC.
2. Launch the Device Manager utility and execute the below action depending on your use case:
  - The STM32CubeProgrammer DFU driver is not installed:  
Execute the STM32 Bootloader.bat script to install it (see [Preparing the USB serial link for flashing](#)).

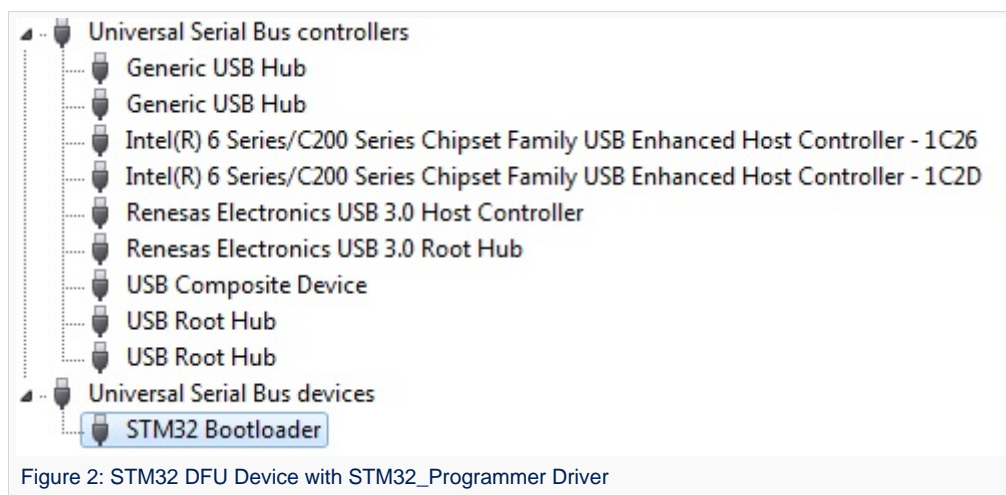


- The DfuSe driver is installed and you must uninstall it.  
Right click **STM Device in DFU Mode** and select **Uninstall**.





- The STM32CubeProgrammer DFU driver is installed and ready to be used.



## 5.4 How to find the DEVICE\_PORT\_LOCATION parameter value for the USB link

**Prerequisites:** STM32CubeProgrammer installation completed and STM32 board connected to the host PC.

To find the value of the DEVICE\_PORT\_LOCATION corresponding to the USB link, use the following command:

```
PC $> STM32_Programmer_CLI -l usb
-----
                STM32CubeProgrammer <tool version>
-----

Total number of available STM32 device in DFU mode: 1

Device Index      : USB1
USB Bus Number    : 002
```



```

USB Address Number : 008
Product ID         : USB download gadget@Device ID /0x500, @Revision ID /0x0000
Serial number      : 00000000000
Firmware version   : 0x0110
Device ID         : 0x0500
  
```

You can then report the value returned by the command line as explained in the [How to flash with STM32CubeProgrammer](#) chapter. Pay attention that the value must be written in **lower case**.

## 5.5 How to explore all the partitions of a populated microSD card

A populated microSD card can contain more than eight partitions, exceeding the number of partitions allowed by default in a Linux system.

At insertion step, you may get an 'Error mounting /dev/mmcblk0p9' or 'mount: cannot mount /dev/mmcblk0p9 read-only' error message.



By adding some options to modprobe, it is possible to allow more than eight partitions on a storage device:

```

PC $> echo 'options mmc_block perdev_minors=16' > /tmp/mmc_block.conf
PC $> sudo mv /tmp/mmc_block.conf /etc/modprobe.d/mmc_block.conf
  
```

## 5.6 How to update partitions

For partial Flash memory update, the flashlayout.tsv must be edited to select only the partition that needs to be updated (first column =1).

See [STM32CubeProgrammer\\_flashlayout#Updating partitions](#) for details and example, the rest of the procedure is the same.

On ST boards, if FSBL or SSBL partitions need to be updated, the Bin2boot fields must be configured with **ST binaries**, as described in [STM32CubeProgrammer\\_flashlayout#Updating partitions using official ST bootloaders](#).

## 5.7 How to populate a microSD card inserted in a Linux host PC

STMicroelectronics distribution is delivered with the [create\\_sdcard\\_from\\_flashlayout.sh](#) script that allows to prepare and flash a microSD card image.

## 5.8 How to fuse STM32MP15x OTP

### 5.8.1 Connection

STM32CubeProgrammer requires to be connected to U-Boot in order to access [OTP alternate](#).



To do it, you have to initiate a flash process but selecting only the 2 first "boot" partitions TF-A( 0x1) and U-Boot( 0x03) of a .tsv.

Using STM32CubeProgrammer GUI this can done without editing by using any available .tsv but selecting only the 2 first partitions thanks to "Select" column tick boxes on the left, as follow:

The screenshot shows the STM32CubeProgrammer GUI. The 'Files' tab is active, displaying a table of partitions. The 'Select' column has checkboxes for 0x1 and 0x3. The 'Log' window shows a successful flash operation for partition 0x24.

Select	Opt	Id	Name	Type	IP	Offset	Binary
<input checked="" type="checkbox"/>	-	0x1	fsbl1-boot	Binary	none	0x00000000	tf-a-stm32mp157c-dk2-trusted.stm32
<input checked="" type="checkbox"/>	-	0x3	ssbl1-boot	Binary	none	0x00000000	u-boot-stm32mp157c-dk2-trusted.stm32
<input type="checkbox"/>	P	0x4	fsbl1	Binary	mmc0	0x00004400	tf-a-stm32mp157c-dk2-trusted.stm32
<input type="checkbox"/>	P	0x5	fsbl2	Binary	mmc0	0x00044400	tf-a-stm32mp157c-dk2-trusted.stm32
<input type="checkbox"/>	P	0x6	ssbl1	Binary	mmc0	0x00084400	u-boot-stm32mp157c-dk2-trusted.stm32
<input type="checkbox"/>	P	0x21	bootfs	System	mmc0	0x00284400	st-image-bootfs-openstlinux-weston-
<input type="checkbox"/>	P	0x22	vendorfs	FileSystem	mmc0	0x04284400	st-image-vendorfs-openstlinux-westo
<input type="checkbox"/>	P	0x23	rootfs	FileSystem	mmc0	0x05284400	st-image-weston-openstlinux-weston-
<input type="checkbox"/>	P	0x24	userfs	FileSystem	mmc0	0x33c84400	st-image-userfs-openstlinux-weston-

The log window shows the following output:

```

05:30:35 : PartID: :0x24
05:30:36 : Start operation done successfully at partition 0x24
05:30:36 : Flashing service completed successfully
09:01:45 : Warning: Connection to device 0x500 is lost
09:01:47 : Disconnected from device.
09:03:11 : Error: Device has been disconnected
09:03:16 : USB speed : High Speed (480Mbit/s)
09:03:16 : Manuf. ID : STMicroelectronics
09:03:16 : Product ID : DPU in HS Mode @Device ID /0x500, @Revision ID /0x0000
09:03:16 : SN : 00250025333851053438330
09:03:16 : FW version : 0x0110
09:03:16 : Device ID : 0x0500

```

Perform the "Download" till success popup then disconnect from the GUI and switch to CLI in a terminal without rebooting the board:

```
PC $> STM32_Programmer_CLI -c port=usb1
```

### 5.8.2 Display status

It will also displayed the status for each OTP words (Shadow lock read/write, program stick lock, permanent write lock):

- All lower OTP
- Only upper that are available for non secure (defined in TF-A device tree).

To display the OTP:

```
PC $> STM32_Programmer_CLI -c port=usb1 -otp displ
```

```
-----
STM32CubeProgrammer v2.3.0
-----
```



```

USB speed      : High Speed (480MBit/s)
Manuf. ID     : STMicroelectronics
Product ID    : USB download gadget@Device ID /0x500, @Revision ID /0x0000
SN            : 002500253338510534383330
FW version    : 0x0110
Device ID     : 0x0500
Device name   : STM32MPxxx
Device type   : MPU
Device CPU    : Cortex-A7

```

```

UPLOADING OTP STRUCTURE ...
  Partition    : 242
  Size         : 1024 Bytes

```

```

Uploading OTP data:
[=====] 100%

```

OTP Partition read successfully

```

OTP STRUCTURE VERSION      :0x00000001

```

BSEC CONTROL REGISTERS:

```

BSEC_OTP_CONFIG           :0x0000000e
  |_[08] TR[1]              : 0
  |_[07] TR[0]              : 0
  |_[06] PRGWIDTH[3]        : 0
  |_[05] PRGWIDTH[2]        : 0
  |_[04] PRGWIDTH[1]        : 0
  |_[03] PRGWIDTH[0]        : 1
  |_[02] FRC[1]             : 1
  |_[01] FRC[0]             : 1
  |_[00] PWRUP              : 0

```

```

BSEC_OTP_Status           :0x00000041
  |_[07] BIST2LOCK          : 0
  |_[06] BIST1LOCK          : 1
  |_[05] PWRON              : 0
  |_[04] PROGFAIL           : 0
  |_[03] BUSY                : 0
  |_[02] INVALID            : 0
  |_[01] FULLDBG            : 0
  |_[00] SECURE             : 1

```

```

BSEC_OTP_LOCK             :0x00000002
  |_[04] GPLOCK             : 0
  |_[03] FENREG             : 0
  |_[02] DENREG             : 0
  |_[00] OTP                 : 0

```

```

BSEC_DENABLE              :0x0000047f
  |_[10] DBGSWENABLE        : 1
  |_[09] CFGSDISABLE        : 0
  |_[08] CP15SDISABLE[1]    : 0
  |_[07] CP15SDISABLE[0]    : 0
  |_[06] SPNIDEN            : 1
  |_[05] SPIDEN             : 1
  |_[04] HDPEN              : 1
  |_[03] DEVICEEN           : 1
  |_[02] NIDEN              : 1
  |_[01] DBGEN              : 1
  |_[00] DFTEN              : 1

```

```

BSEC_FENABLE              :0x00000000

```



```

|_[03] CAN_disable           : 0
|_[02] GPU_disable          : 0
|_[01] Dual_A7_disable      : 0
|_[00] Crypto_disable       : 0

```

## OTP LOWER SHADOW REGISTERS:

ID	value	disturbed	error	R_SLock	W_SLock	Prog_SL	Perm_L
00	0x00000017	0	0	1	1	0	0
01	0x00008000	0	0	0	1	0	1
02	0x00000000	0	0	0	1	1	0
03	0x00000000	0	0	0	0	0	0
04	0x00000000	0	0	0	0	0	0
05	0x00000000	0	0	0	0	0	0
06	0x00000000	0	0	0	0	0	0
07	0x00000000	0	0	0	0	0	0
08	0x00000000	0	0	0	1	0	0
09	0x00000000	0	0	0	0	0	0
10	0x00000000	0	0	0	0	0	0
11	0x00000000	0	0	0	0	0	0
12	0x7ce7f0ee	0	0	0	0	0	1
13	0x00250025	0	0	0	0	0	1
14	0x33385105	0	0	0	0	0	1
15	0x34383330	0	0	0	0	0	1
16	0x125575aa	0	0	0	1	1	1
17	0x1f41185b	0	0	0	0	0	1
18	0x7a200140	0	0	1	1	0	0
19	0x0690147c	0	0	0	0	0	1
20	0x5de00048	0	0	0	0	0	1
21	0x00000000	0	0	0	0	0	1
22	0x00000000	0	0	0	0	0	1
23	0x3fec2fd7	0	0	0	0	0	1
24	0x00000000	0	0	0	1	0	0
25	0x00000000	0	0	0	1	0	0
26	0x00000000	0	0	0	1	0	0
27	0x00000000	0	0	0	1	0	0
28	0x00000000	0	0	0	1	0	0
29	0x00000000	0	0	0	1	0	0
30	0x00000000	0	0	0	1	0	0
31	0x00000000	0	0	0	1	0	0

## OTP UPPER SHADOW REGISTERS:

ID	value	disturbed	error	R_SLock	W_SLock	Prog_SL	Perm_L
32	0x00000000	0	0	1	1	1	1
33	0x00000000	0	0	1	1	1	1
34	0x00000000	0	0	1	1	1	1
35	0x00000000	0	0	1	1	1	1
36	0x00000000	0	0	1	1	1	1
37	0x00000000	0	0	1	1	1	1
38	0x00000000	0	0	1	1	1	1
39	0x00000000	0	0	1	1	1	1
40	0x00000000	0	0	0	1	1	1
41	0x00000000	0	0	0	1	1	1
42	0x00000000	0	0	0	1	1	1
43	0x00000000	0	0	0	1	1	1
44	0x00000000	0	0	0	1	1	1
45	0x00000000	0	0	0	1	1	1
46	0x00000000	0	0	0	1	1	1
47	0x00000000	0	0	0	1	1	1
48	0x00000000	0	0	0	1	1	1
49	0x00000000	0	0	0	1	1	1
50	0x00000000	0	0	0	1	1	1
51	0x00000000	0	0	0	1	1	1





52	0x00000000	0	0	0	1	1	1
53	0x00000000	0	0	0	1	1	1
54	0x00000000	0	0	0	1	1	1
55	0x00000000	0	0	0	1	1	1
56	0x00000000	0	0	1	1	0	0
57	0x42e18000	0	0	0	0	0	0
58	0x0000fd51	0	0	0	0	0	0
59	0x12722301	0	0	0	0	0	0
60	0x00000000	0	0	0	0	0	0
61	0x00000000	0	0	0	0	0	0
62	0x00000000	0	0	0	0	0	0
63	0x00000000	0	0	0	0	0	0
64	0x00000000	0	0	0	0	0	0
65	0x00000000	0	0	0	0	0	0
66	0x00000000	0	0	0	0	0	0
67	0x00000000	0	0	0	0	0	0
68	0x00000000	0	0	0	0	0	0
69	0x00000000	0	0	0	0	0	0
70	0x00000000	0	0	0	0	0	0
71	0x00000000	0	0	0	0	0	0
72	0x00000000	0	0	0	0	0	0
73	0x00000000	0	0	0	0	0	0
74	0x00000000	0	0	0	0	0	0
75	0x00000000	0	0	0	0	0	0
76	0x00000000	0	0	0	0	0	0
77	0x00000000	0	0	0	0	0	0
78	0x00000000	0	0	0	0	0	0
79	0x00000000	0	0	0	0	0	0
80	0x00000000	0	0	0	0	0	0
81	0x00000000	0	0	0	0	0	0
82	0x00000000	0	0	0	0	0	0
83	0x00000000	0	0	0	0	0	0
84	0x00000000	0	0	0	0	0	0
85	0x00000000	0	0	0	0	0	0
86	0x00000000	0	0	0	0	0	0
87	0x00000000	0	0	0	0	0	0
88	0x00000000	0	0	0	0	0	0
89	0x00000000	0	0	0	0	0	0
90	0x00000000	0	0	0	0	0	0
91	0x00000000	0	0	0	0	0	0
92	0x00000000	0	0	0	0	0	0
93	0x00000000	0	0	0	0	0	0
94	0x00000000	0	0	0	0	0	0
95	0x00000000	0	0	0	0	0	0

```

BSEC_HWCFGR :0x00000014
|_[07] ECC_USE[3] : 0
|_[06] ECC_USE[2] : 0
|_[05] ECC_USE[1] : 0
|_[04] ECC_USE[0] : 1
|_[03] SAFMEM_SIZE[3] : 0
|_[02] SAFMEM_SIZE[2] : 1
|_[01] SAFMEM_SIZE[1] : 0
|_[00] SAFMEM_SIZE[0] : 0
    
```

BSEC IP REV :1.1

BSEC IP ID :0x00100032

BSEC IP MAGIC ID :0xa3c5dd04

### 5.8.3 Update value

It is now possible to use programmer to manage new fuses values: bit to bit for Lower, only programmable once for Upper.



For example: Set the value **0x21458000** into the word **57 (0x39)**

```
PC $> STM32_Programmer_CLI -c port=usb1 -otp program wordID=0x39 value=0x21458000
```

Words are written as a 32bit word. It is important to take care of the bit order once using the value in software.

Example: Mac address 00:80:e1:42:45:e5 must be written as:

- Word 57 : 0x42e18000
- Word 58 : 0x0000e545

### Warning

MAC address / board Id OTP write should not be performed on ST boards ! They are written and protected by ECC during factory production. Any further write will break the board with no way to recover if these OTP are not locked

## 5.8.4 Update lock

Lock (no other programmation could occurred) could be added per word; the permanent lock access will only be refreshed after boot.

For example: permanent lock (**pl=1**) the word **57 (0x39)**

```
PC $> STM32_Programmer_CLI -c port=usb1 -otp program wordID=0x39 value=0x21458000 pl=1
```

## 5.9 How to program STPMIC NVM

STM32CubeProgrammer require to be connected to U-Boot in order to access STPMIC NVM alternate (0xF4).

Refer to previous Chapter 5.8.1 and follow same procedure to flash only the 2 first "boot" partition with GUI. Then disconnect from GUI and switch to CLI in a terminal.

This is procedure to modify NVM values.

- Read partition of pmic and load it into a .bin file

```
PC $> STM32_Programmer_CLI -c port=usb1 -rp 0xf4 0x0 0x8 $MySelectedPATH\PMIC_NVM_read.  
bin
```

- Backup it by creating a copy PMIC\_NVM\_write.bin
- Use a binary editor in HEX format ( eg "vi -b file.bin" then :%!xxd ) to edit the copy PMIC\_NVM\_write.bin with new values.

The binary format display the shadow register from 0xF8 to 0xFF. Refer to STPMIC1 DataSheet DS12792.

```
00000000: EE92 C002 F280 0233 .....3  
respectively SHR address : F8F9 FAFB FCFD FEFF
```

- Program back the STPMIC1 with modified NVM using following command:

```
PC $> STM32_Programmer_CLI -c port=usb1 -pmic "PMIC_NVM_write.bin"
```



Universal Asynchronous Receiver/Transmitter

former spelling for e•MMC ('e' in italic)

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Device Firmware Upgrade

Operating System

High Speed (MIPI<sup>®</sup> Alliance DSI standard)

Microprocessor Unit

System Trace Module

First Stage Boot Loader

Second Stage Boot Loader

One Time Programmed

Das U-Boot -- the Universal Boot Loader (see [U-Boot\\_overview](#))

Trusted Firmware for Arm<sup>®</sup> Cortex<sup>®</sup>-A

Graphical User Interface

Central processing unit

Cortex<sup>®</sup>

Boot and Security and OTP control

Controller Area Network (robust bus mainly used for automotive applications)

Graphics Processing Units

Elliptic curve cryptography

Error Correction Capability

media access control address ([https://en.wikipedia.org/wiki/MAC\\_address](https://en.wikipedia.org/wiki/MAC_address))

Non Volatile Memory, like a flash memory

Power Management Integrated Circuit