DC11652
Driver 3.11
July 2024
F. Catto

eMMC Flashing with FlashRunner NXG and FlashRunner High-Speed

1. Introduction

This document aims to explain the flashing of eMMC memories with FlashRunner NXG (NXG) and FlashRunner High Speed (FR-HS), with a performance comparison between the three platforms. FlashRunner NXG can program eMMC devices, but due to technological limits, it is not able to have a flashing time as competitive as needed by the market. This is one of the reasons that led to the development of the FlashRunner High-Speed platform which, with its innovative technology, is able to reach higher performance in the flashing of big-size memories.

eMMC devices are becoming more and more common in the new high-technological projects, in particular in the Automotive field, and they often determine the flashing time of a panel. So, being able to optimize flashing operations for them, means that we can reduce the flashing time for most of our customers' applications of this kind.

Basically, the driver supports any eMMC memory from the most popular silicon producers, such as: **Cypress**, **ISSI**, **Kioxia**, **Macronix**, **Micron**, **Samsung, Sandisk** and many others. Moreover, if you don't find your code in our device list, it is always possible for us to support a new eMMC, just check by writing an email to the SMH Technologies sales team.

You can download the latest version of this document from this static link: [eMMC Flashing with FlashRunner NXG and FlashRunner High-Speed](#).

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

2. Contents

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

3. Standard flashing commands

In this section, we explain the standard flashing operations that can be executed on the target device and how these operations work. This knowledge may help you to optimize even more your application.
These are the standard commands:

- **Masserase**
  This command erases all the content of the selected partition of the eMMC memory (all bits are set to 0).
  This operation is not strictly mandatory since the memory cells value can change from 0 to 1, but also vice versa; there is an implicit erase executed by the device as part of the write operation. It is strongly recommended to execute a masserase before reprogramming a target.

  The duration of this operation is typically very low (less than 5 seconds), because it doesn't erase physically the memory cells, but it moves the mapped host address range to the unmapped host address range of the memory controller. This way the memory cells result erased when read by the user.

- **BlankCheck**
  This command reads all the content of the selected partition of the eMMC memory and checks that all bits are set to 0. This operation is terminated instantly when a 1 is found.
  Optionally, it is also possible to send two additional parameters to this command: the address from where to start reading and the number of bytes to check.

  This operation is not strictly mandatory since the memory cell value can change from 0 to 1, but also vice versa; there is an implicit erase executed by the device as part of the write operation. This command, since it is not mandatory, is in the "Additional Commands" window during the creation of the project using the Workbench.

  The duration of this operation depends on the bitrate that has been set by the user. The bit rate directly depends on the protocol and the frequency, so the suggestion is to use the highest frequency and the fastest protocol. See the examples below which show the time estimation for the blankcheck operation over a 1GB serial eMMC flash memory:

Protocol	Frequency	Bitrate	Time for 1GB
eMMC-1bit	8 MHz	1 MBps	1024 s
eMMC-4bit	24 MHz	12 MBps	85 s
eMMC-8bit	50 MHz	45 MBps	23 s

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

- **Program**
  This command takes the customer's data from the FRB file and programs them into the memory.
  Optionally, it is also possible to send two additional parameters to this command: the address from where to start programming and the number of bytes to be programmed.

  The duration of this command mainly depends on the bitrate. FlashRunner sends the data in blocks (512 bytes) to the device. So, having a higher bitrate reduces the "send block" time. The reduction of the program operation is inversely proportional to the increase of the bit rate.

  In addition to the increase of the bitrate, it's possible to activate some optimization over the FRB file to improve the performance:
  1. During the creation of the FRB file, in the "FRB Management" section, activate the option ***"Remove Blank Pages from source file during FRB conversion [30KB page size]"***. With this option, all the blank areas in the source file (with a page size of 30KB) are ignored and not included in the FRB file. This way the amount of data sent to the device is reduced.

  2. Only for FlashRunner NXG: set the FRB file with the "IGNORE_BLANK_PAGE" option. This will skip the program operation for any page which contains only 0xFF bytes. Anyway, the suggested solution is the previous one.

     #TPSETSRC myData.frb *IGNORE_BLANK_PAGE*

- **Verify**
  This command reads all the content of the eMMC memory and checks that it corresponds to FRB data. This operation is terminated instantly when a mismatch is found.
  Optionally, it is also possible to send two additional parameters to this command: the address from where to start reading and the number of bytes to check.

  The duration of this operation should be the same as blankcheck, but with a very small delay because we need to manage and check data.
  To improve the duration of this command, follow the same suggestions proposed for blankcheck and program commands.

After this explanation should be clear what can be improved by the user, what can be improved by SMH, and what cannot be improved because it depends only on the characteristics of the target device.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

4. Protocols and frequencies supported

The FlashRunner supports three different protocols for the flashing:

1. **eMMC-1bit**: one data line is used (DAT0). A single channel is used for each eMMC and it's possible to mix up different protocols and technologies also with FlashRunner NXG.

2. **eMMC-4bit**: four data lines are used (DAT0/1/2/3). A single channel is used for each eMMC and it's possible to mix up different protocols and technologies also with FlashRunner NXG.

3. **eMMC-8bit**: eight data lines are used (DAT0/1/2/3/4/5/6/7). With FlashRunner NXG this means that two channels have to be used to program a single eMMC and, due to its technological limits, it's not possible to mix up different protocols and technology: all the channels have to program eMMC devices. This limitation is not present with FlashRunner HS.

The frequencies supported are different from FlashRunner NXG and FlashRunner HS:

- **FlashRunner NXG**: up to **25 MHz**. Due to a technological bottleneck, even increasing the communication speed of the protocol beyond this value, the overall bitrate doesn't increase but only signal integrity problems might arise. All communication protocols (see above) available.
- **FlashRunner HS**: up to **75 MHz**. Thanks to the dedicated Active Module for the eMMC flashing is possible to reach this frequency that converts to a bitrate of 70MBits. All communication protocols (see above) available.

Attention: if you notice some instabilities when rising frequency or when trying to use eMMC-4bit or eMMC-8bit, it could be related to the hardware setup (see the next chapter).

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

5. Hardware setup

Just like an F1 car, which differently performs according to the weather and track conditions, the FlashRunner requires a good hardware setup to reach the best performance. The quality of the connections between the FlashRunner and the target board is extremely important. In this chapter we want to define some guidelines.

The test points needed for each protocol are:

Protocol	DIO0	DIO1	DIO2	DIO3	DIO4	DIO5	DIO6	DIO7	DIO8	DIO9	DIO10	VPROG0	VPROG1	GND
eMMC-1bit	RST	CLK	CMD	DAT0	-	-	-	-	-	-	-	VCCq	VCC	GND
eMMC-4bit	RST	CLK	CMD	DAT0	DAT1	DAT2	DAT3	-	-	-	-	VCCq	VCC	GND
eMMC-8bit	RST	CLK	CMD	DAT0	DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	DAT7	VCCq	VCC	GND

Notes:

1. For FlashRunner NXG, DIO8, DIO9 and DIO10 are on the second channel, respectively at: DIO0, DIO1, DIO2.
2. The RST line is not mandatory. For backward compatibility of the eMMC standard, this signal is disabled by default. The host may set it as permanently enabled or disabled by writing the appropriate value in the RST_n_FUNCTION extended CSD register.
3. VPROG0 and VPROG1 don't need to be connected if the board is powered using a dedicated power supply.
4. For FlashRunner NXG in eMMC-8bit mode, the grounds of both channels used have to be connected.

The most recent eMMC devices all have a Dual Voltage power supply: Vccq for the eMMC controller core and VCC for the NAND interface. To ensure higher signal integrity it is suggested to set the VCC to 3.3 V and the VCCq to 1.8 V. This means to set the VPROG0 to 1.8 V and, if used to power the eMMC, the VPROG1 to 3.3 V (this value is possible only for FlashRunner High-Speed, with FlashRunner NXG the customer has to provide the 3.3 V with an external power supply due to VPROG1 range limits).

As explained in the previous chapter, using eMMC-4bit or eMMC-8bit gives a big advantage because the bitrate can be much higher, but there is also a disadvantage: a good hardware setup is required. In fact, while eMMC-1bit is very robust and can work also in bad conditions, eMMC-4bit and eMMC-8bit are more sensitive because they have many synchronous data lines and the quality of the wirings is crucial:

- With FlashRunner NXG, use our cable interface to go as close as possible to the target device.

With FlashRunner High-Speed, place the eMMC Active Module as close as possible to the target.

- The wiring should be as short as possible.
- All wiring should have the same length.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

- There should be as few discontinuities as possible (i.e. prefer one 30 cm cable instead of two 15 cm cables connected together).
- With FlashRunner High Speed, use a maximum connection between the Active Module and the target of 20/30 cm to exploit at best the technology.
- Ideally, one ground line for each DIO should be connected. On the board there would not be a test point for each ground. These lines should be grouped together on the GND test points close to the target.
- Twist all the DIOs with the relative ground.

There are also some PCB hints that we can suggest:

- Series termination resistors are placed as close as possible to the target device.
- Series termination resistors value in the range of 10 Ω to 27 Ω according to the target frequency (the higher, the smaller the resistance).
- Test points are placed directly on the route to minimize the “stub effect”.
- Test points placed before the series resistor. The path should be: test point, series resistor, device.
- Test points size as small as possible according to nails/probes features.
- Single-ended line 50 Ω controlled impedance (5% of tolerance)
- Clock and data communication line length matching the distance between device pins and test points.

A common issue that can arise when multiple lines move at the same time is crosstalk: the phenomenon by which signals are transmitted on one or more lines generates undesired effects on other lines due to electromagnetic coupling between them. For example, when transmitting 0x00 followed by 0xFF in eMMC-8bit, the simultaneous switching of eight data lines from 0 to 1 can pull the clock line to 1 and generate a spurious clock pulse.
This effect may arise if no series resistor has been placed on the line or the test point is after the resistor, this means the path is: series resistor (or no resistor), test point, device. In this case, one possible solution is to add small resistors (e.g. 20 Ω) in series externally and as close as possible to the test point, on all data lines and the command line. These resistors decrease the current flow on data lines and then reduce the interaction effect on the clock line.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

6. Partitioning and partitions settings

eMMC devices have three default partitions:

1. F: User Data Area. This is the main area partition where the data of the user will be stored. The address range is from 0x00 until the end of the memory.
2. B, C: Boot Partition 1 and 2. These areas are separated from the user data area. The start address is from 0x00 until the end of the relative boot partition.

User Data Area, by default, is implemented as Multi-Level Cells (MLC); this means that for each data cell two bits are stored. This solution is used to increase memory density, but it decreases the reliability of the storage. The boot partitions, instead, are typically implemented as Single-Level Cells (SLC), where a single bit is stored for each cell increasing, this way, the write performances and the reliability. A portion of the User Data Area can be set in SLC, reducing the overall memory size, with the Enhanced attribute:

#TPCMD SET_ENHANCED_AREA <MAX|start_addr> <size>

- *MAX*: create an Enhanced User Data Area from address 0x00 and the maximum size allowed by the device.
- *start_addr*: the start address of the Enhanced Area inside the User Data Area.
- *size*: the size of the Enhanced Area. The size has to be aligned to the HIGH CAPACITY WRITE PROTECT GROUP SIZE; if not aligned, the driver will change it to the smallest aligned size to fit the size requested by the command. The device has a maximum enhanced area allowed, it is not possible to set the whole User Data Area as SLC.

In addition to these partitions is possible to set up 4 **General-Purpose Partitions** (GP1, GP2, GP3, and GP4). These are Hardware Partitions that are set by writing OTP extended CSD registers; once set the partitions cannot be removed or changed. These partitions "take" their space from the User Data Area, this means that after the partitioning the User Data Area is smaller. All the partitions have the start address at 0x00. An alternative to the hardware partition is the software partition which has to be managed by the customer's firmware with an MBR or GPT table. To create the General Purpose Partitions the user can use the following command:

#TPCMD *CREATE_PARTITION <size> <enhanced_setting> <extended_setting>*

- *size*: the size of the partition. It has to be aligned to the *HIGH_CAPACITY_WRITE_PROTECT_GROUP_SIZE*; if not aligned, the driver will change it to the smallest aligned size to fit the size requested by the command.
- *enhanced_setting*(optional): a general-purpose partition can have enhanced area attribute. To enable this feature set this field to 1, otherwise it can be ignored. *NOTE:* if the partition has an extended attribute, set this field to 0. Remember that a partition can be Enhanced *or* Extended, not both. Moreover, the device has an overall maximum enhanced area allowed.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

- *extended_setting*(optional): a general-purpose partition can have an extended attribute. This is used to "mark" the partition to have a specific purpose. If not needed, this field can be ignored. The possible options are:
    - 0x00: no extended attribute (default).
    - 0x01: system code. Partition rarely updated and contains system files.
    - 0x02: non-persistent. Partition used for temporary information.

Here below is a table with all the possible combinations for this command:

***#TPCMD** CREATE_PARTITION*	<size>	<enhanced_setting>	<extended_setting>	
		-	-	simple partition (default)
		0x00	0x00	
		0x01	0x00	enhanced
		0x01		
		0x00	0x01	system code
		0x00	0x02	non-persistent

The enhanced User Data Area and the General-Purpose partitions are **OTP** settings. To save the configuration the user needs to send the command:

***#TPCMD** SAVE_PARTITION*

After this command is sent, the device makes an internal check of the setting inserted and, if something is not correct, it returns an error. At the next power-up it is possible to set again the partitions. To enable the partitioning the device has to perform a Hardware Reset, i.e. powering down and then powering up the device. This can be done automatically with the DISCONNECT command if the device is powered by the FlashRunner. Otherwise, if it is powered by an external power supply, the user has to manage the power cycle. After the reset, the device will take a longer time to initialize itself, but this happens only the very first time after the partitioning.

Here below a script example for a SDINBDG4-32G, which is a 32GB eMMC:

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
#TPSTART
#TPCMD CONNECT
#TPCMD CREATE_PARTITION 0x100000000 0x01 0x00
#TPCMD CREATE_PARTITION 0x200000000 0x00 0x01
#TPCMD CREATE_PARTITION 0x100000000 0x00 0x02
#TPCMD CREATE_PARTITION 0x100000000
#TPCMD SET_ENHANCED_AREA 0x100000000 0x100000000
#TPCMD SAVE_PARTITION
#TPCMD DISCONNECT
#TPCMD CONNECT
;... program
#TPCMD DISCONNECT
#TPEND
```

When the partitions are created, at each connect in the log are printed the device information:

```
21|1|eMMC user data size: 9340 MB, 0x0247C00000 (9793699840) Bytes
21|1|General Purpose Partition 1 size: 0x0100000000 Bytes with Enhanced attribute
21|1|General Purpose Partition 2 size: 0x0200000000 Bytes with Extended attribute 0x01
21|1|General Purpose Partition 3 size: 0x0100000000 Bytes with Extended attribute 0x02
21|1|General Purpose Partition 4 size: 0x0100000000 Bytes
21|1|Enhanced User Data Area start address: 0x100000000 - size: 0x100000000 Bytes
21|1|eMMC total memory size: 0x0747C00000 (31268536320) Bytes
```

When the eMMC is partitioned, all the partitions have as start address 0x00. To program each partition the user has to provide the proper image file to be converted in FRB and then set it as source file before programming the partition. The FRB files can be generated using the "Advanced FRB Manager" and selecting the correct device; remember to set the *"Remove Blank Pages from source file during FRB conversion [30KB page size]"* option to increase the performance, especially if you are using the FlashRunner HS. Here below an example of the flow of the flashing:

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

```
#TPCMD CONNECT
#TPSETSRC GP1.frb
#TPCMD PROGRAM GP1
#TPCMD VERIFY GP1 R
#TPSETSRC GP2.frb
#TPCMD PROGRAM GP2
#TPCMD VERIFY GP2 R
#TPSETSRC UserDataArea.frb
#TPCMD PROGRAM F
#TPCMD VERIFY F R
#TPSETSRC Boot1.frb
#TPCMD PROGRAM B
#TPCMD VERIFY B R
;... other operations
#TPCMD DISCONNECT
```

#TPCMD *RESIZE_BOOT_PARTITIONS <BOOT_SIZE_MULTIPLIER> <RPMB_SIZE_MULTIPLIER>*

This command enables larger BOOT and RPMB partitions than what is specified in the standards, but it is only available on a limited number of eMMC devices.
After sending this command, the device will adjust the size of the Boot and RPMB partitions based on the provided multipliers.

BOOT_SIZE_MULT = 0x20 or up to 0xFF if resized
Size = BOOT_SIZE_MULT * 128 KB

RPMB_SIZE_MULT = 0x20 or up to 0x80 if resized
Size = RPMB_SIZE_MULT * 128 KB

To make the change effective, power cycle the memory.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

7. Extended CSD Registers

The Extended CSD register, 512 bytes long, defines the device properties and selected modes. The FlashRunner uses this register to retrieve the necessary information to work properly, like: the presence of partitions, their size …

The driver allows the user to get the content of this register with the command:

***#TPCMD** READ_EXTCSD <extCSD_index>*

- extCSD_index: the index of the CSD-slice the user wants to get. For example, index 179 is for the PARTITIONING_CONFIG which contains the configuration of the partitions of the device.

If no index is inserted, all the 512 bytes of the register are returned.

It is also possible for the user to a value for a specific field of the Extended CSD register:

***#TPCMD** WRITE_EXTCSD <extCSD_index> <value>*

- extCSD_index: the index of the CSD-slice the user wants to write.
- Value: the byte the user wants to program

Please take into consideration that some configurations are OTP, therefore the user has to take care of this aspect.

Note: the index parameter must be taken from the documentation of the target device.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

8. INFO command

The user can get from the driver some useful information and print them on the log and on the communication channel with the DLL. The following command can be used:

#TPCMD *GET_INFO <info_name> <field_name>*

The following *info_name* can be used:

- **TOTAL_MEMORY_SIZE**: the total memory size in hex value.
- **USER_DATA_AREA_SIZE**: the user data area size in hex value.
- **USER_ENHANCED_AREA_SIZE**: the user enhanced area size in hex value.
- **USER_ENHANCED_AREA_START_ADDRESS**: the user-enhanced area start address in hex value.
- **CID_REGISTER**: *field_name*:
  - The specific field of the register. Example: MID, OID …
  - *ALL*: print all the fields of the register
  - *Not set*: print the full CID register
- **CSD_REGISTER**:
  - The specific field of the register. Example: C_SIZE, WP_GRP_SIZE …
  - *ALL*: print all the fields of the register
  - *Not set*: print the full CSD register
- **EXT_CSD_REGISTER**:
  - The specific field of the register. Example: BOOT_SIZE_MULT, HC_ERASE_GRP_SIZE …
  - *ALL*: print all the fields of the register

Here below an example of usage:

```
01|2|---#TPCMD GET_INFO CID_REGISTER ALL
01|1|CID: 0x13014E51324A3534411011EE7CC7B7E4
01|1|MID [127:120]: 0x000000000013
01|1|RESERVED [129:114]: 0x000000000000
01|1|CBX [113:112]: 0x000000000001
01|1|OID [111:104]: 0x00000000004E
01|1|PNM [103:056]: 0x51324A353441
01|1|PRV [055:048]: 0x000000000010
01|1|PSN [047:016]: 0x000011EE7CC7
01|1|MDT [015:008]: 0x0000000000B7
01|1|CRC [007:001]: 0x000000000072
01|1|NOT_USED [000:000]: 0x000000000000
```

```
01|2|---#TPCMD GET_INFO CID_REGISTER MID
01|1|CID: 0x13014E51324A3534411011EE7CC7B7E4
01|1|MID [127:120]: 0x000000000013
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

9. Driver parameters (TCSETPAR)

In this chapter is explained the meaning of the driver parameters:

- **#TCSETPAR** CMD_PULLUP <YES|NO>

  This parameter is used to enable the pull-up on the CMD line. The default value, when creating a new project, is YES. This setting can be very useful when the integration setup is not optimal and, for example, not negligible stray capacitance effect is introduced and the eMMC is no longer able to drive properly to the high value the line. This setting introduces a low-pass filter effect which does not allow programming at high frequencies that, anyway, is possible only with a good setup, where the pull-up is not necessary

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

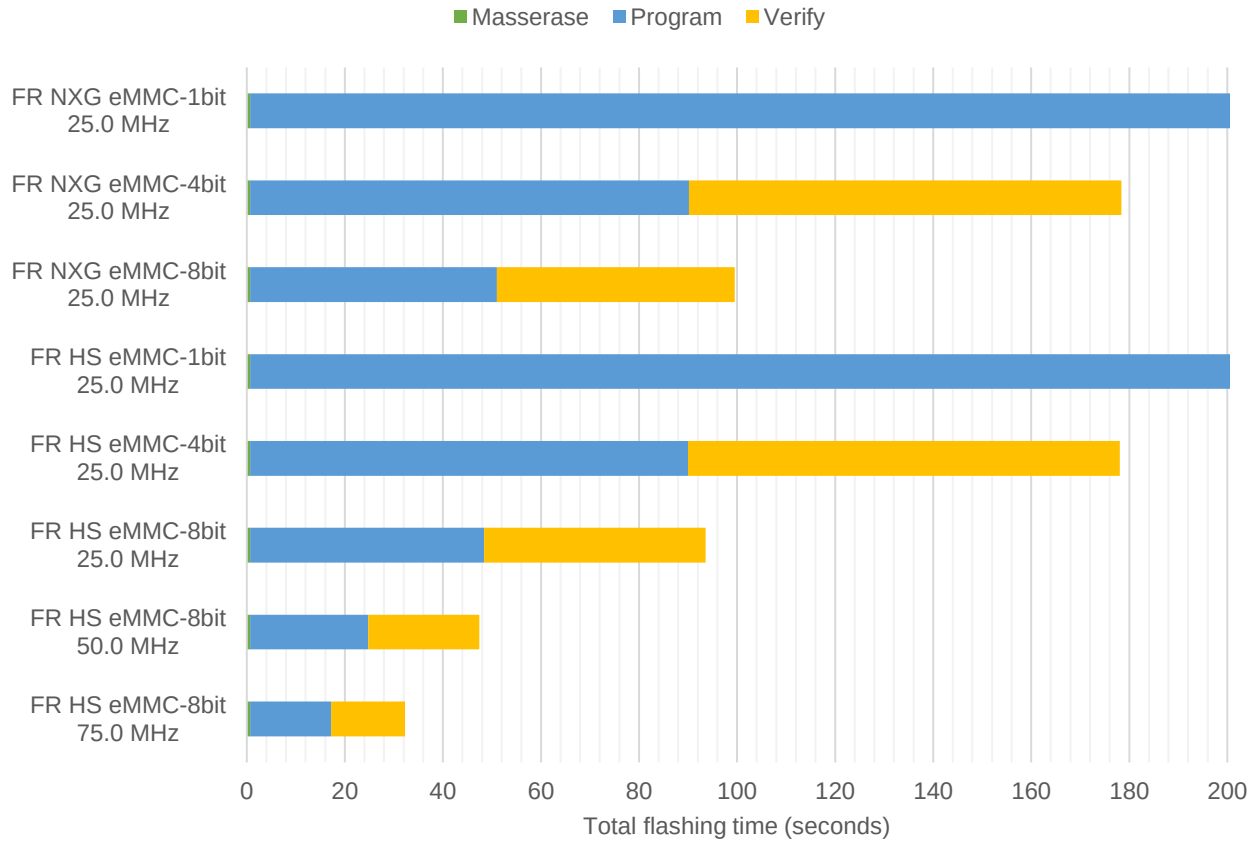
10. Flashing time examples

These are some examples of flashing times using libemmc.so driver.

Additional notes:

1. Times were measured using FRB files containing random data to cover the entire memory of the target device.
2. We do not consider the blankcheck time in the total flashing time because it is usually skipped.

SDINBDA4-32G [1 GB]



Platform	Protocol	Masserase (s)	Program (s)	Verify (s)
FlashRunner NXG	eMMC-1bit 25.0 MHz	0.700	347.100	344.300
FlashRunner NXG	eMMC-4bit 25.0 MHz	0.700	89.500	88.200
FlashRunner NXG	eMMC-8bit 25.0 MHz	0.700	50.300	48.500

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

FlashRunner HS	eMMC-1bit 25.0 MHz	0.700	347.000	345.600
FlashRunner HS	eMMC-4bit 25.0 MHz	0.700	89.300	88.100
FlashRunner HS	eMMC-8bit 25.0 MHz	0.700	47.700	45.200
FlashRunner HS	eMMC-8bit 50.0 MHz	0.700	24.100	22.600
FlashRunner HS	eMMC-8bit 75.0 MHz	0.700	16.500	15.100

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

11. Frequently Asked Questions

Some answers to the most frequently asked questions:

How many channels are needed to use eMMC-8bit?

As mentioned before, eMMC-8bit needs eight data lines, a clock, a command and reset signals. This means eleven DIOs of the FlashRunner NXG and they cannot be managed by a single channel, so eMMC-8bit needs a pair of channels, for example, channels 1 and 2, or 3 and 4, etc.
This implicates some additional limitations on the *FlashRunner NXG*: when using this special mode, the FlashRunner can parallelly flash only devices contained in the eMMC driver and the devices can only be placed starting from the odd channels.
It is not even possible to flash devices which are not eMMCs when using the FlashRunner NXG in eMMC-8bit mode.
Instead, for the *FlashRunner HS*, the situation is easier because there are dedicated active modules which do the job, so it's possible to mix up different technologies. Moreover, it is possible to flash up to 16 eMMCs in parallel.

What is causing verify readout errors on eMMC-4bit or eMMC-8bit?

This is a common problem that happens when trying to use eMMC-4bit or eMMC-8bit with a non-optimal hardware setup. This error is caused by electromagnetic noise and it happens especially when the FlashRunner has to read a big amount of data from the target device. The very first thing to do to fix this problem is to use eMMC-1bit, this should always work if the wiring is correct.
Instead, to keep using eMMC-4bit and eMMC-8bit, one possible solution is lowering the protocol frequency. If it is not working yet, then it means that the hardware setup requires a review so, please, see the chapter "Hardware setup".

How to calculate the HIGH_CAPACITY_WRITE_PROTECT_GROUP_SIZE?

The *HIGH_CAPACITY_WRITE_PROTECT_GROUP_SIZE* is a value used for the alignment of the address and size in the commands. It can be calculated in bytes as:
HIGH_CAPACITY_WRITE_PROTECT_GROUP_SIZE = HC_WP_GRP_SIZE x HC_ERASE_GRP_SIZE x 512 x 1024
The *HC_WP_GRP_SIZE* and *HC_ERASE_GRP_SIZE* are registers of Extended CSD and their value is present in the datasheet of the memory.

What is causing the partitioning error?

When setting the partitions of the eMMC some constraints have to be respected:

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale
Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021
→smh-tech.com
info@smh-tech.com

- The device does not support General Purpose Partition, Enhanced settings or Extended settings. This information is stored in the extended CSD register [160]: *PARTITIONING_SUPPORT*.
- The sum of the General-Purpose Partitions size plus the User Data size has to be less or equal to the size of the device memory.
- A partition can be enhanced **or** extended, not both.
- The size has to be aligned to *HIGH_CAPACITY_WRITE_PROTECT_GROUP_SIZE*. If it's not aligned the driver will assign to the partition the minimum memory big enough to fit the size requested. This way the memory assigned will be bigger and this can lead to an unexpected size violation.
- The device has a maximum enhanced area allowed. This value is: *MAX_ENH_SIZE_MULT x HC_WP_GRP_SIZE x HC_ERASE_GRP_SIZE x 512 x 1024* (see extended CSD registers [159:157], [224] and [221]).
  Hence, this inequation has to be satisfied when setting the enhanced area for a device:
  Σ Enhanced general partition Size(i) + Enhanced user data area ≤ Max Enhanced Area.

Which OS version supports libemmc.so driver?

The libemmc.so driver runs with OS3.xx, the OS2.xx does not support this driver.

Can I use FlashRunner 2.0 to program eMMC?

When we introduced FlashRunner HS we made the decision not to support further eMMC projects with 2.0 since this would have been less performing and, moreover, less suitable for the integration. eMMC programming requires the integration in fixture (eMMC Active Modules or, in some cases, NXG) to guarantee signal integrity and to reach production efficiency. FR2P0 has been a starting point but High Speed is the solution SMH has developed for such kind of applications.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale €102.040
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com