SMH Technologies®

Interfacing FlashRunner with NAND Flash Memories

DC11562
Driver 2.08
September 2023
R. Ertolupi

1. Introduction

This document aims to explain how FlashRunner manages the NAND memories, both the classic parallel NAND and the new serial NAND. First of all, let us explain the peculiarities of these memories.

The increasing request for higher storage capacity, due to the complexity of the new projects, is leading customers to search for bigger memories while keeping the costs as low as possible. NAND flash memories are exactly the solution for these requirements and they are going to replace NOR flash memories for many applications. The reason why NAND flash memories are cheaper than NOR is because of the different architecture, which is simpler and, so, production is cheaper. Moreover, NOR flash memories basically reached the capacity limit because at densities above 512Mbits they scale poorly and their die size and cost are excessive.
On the other hand, NAND flash memories are less reliable than NOR because they can have bitflip and bad blocks. That is why NAND flash memories require error correction code (ECC) and bad block management, which make them less “user-friendly” and a bit “tricky”.
One more advantage of NAND flash memories compared to NOR is the erase and program speed which is much faster. This is extremely important for several applications, such as ADAS, clusters, gaming consoles, systems that require fast over-the-air updates (OTA), autonomous driving systems and artificial intelligence systems in general.

Since classic parallel NAND uses a complex protocol, silicon producers developed new serial NAND flash memories that use Quad-SPI and Octo-SPI protocol (same as the Serial NOR flash memories).

You can download the latest version of this document from this static link: Interfacing FlashRunner with NAND Flash Memories.

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

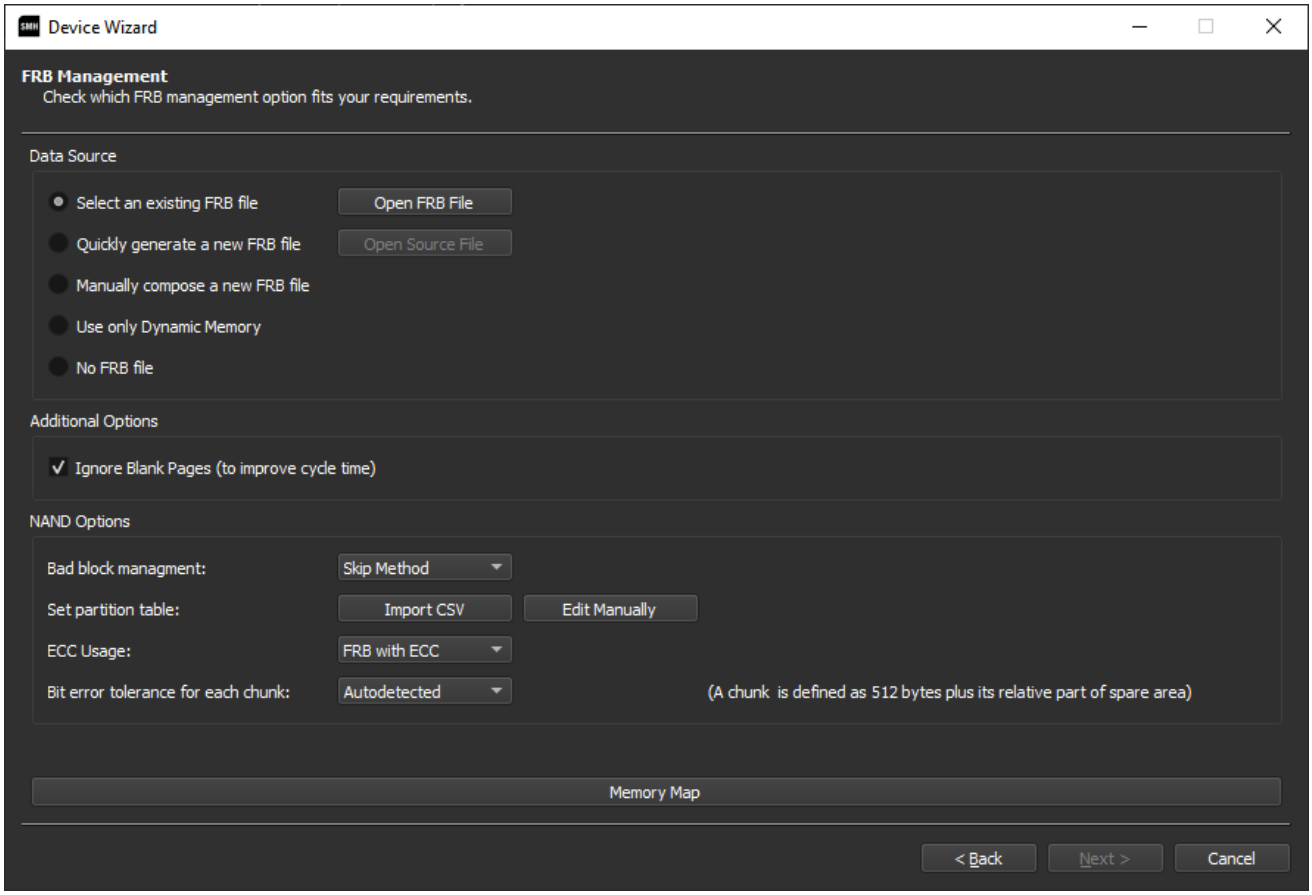
→smh-tech.com

info@smh-tech.com

3. How to create a project?

In this section, we are going to see how to create a brand new project for a NAND memory from our graphical user interface (aka Workbench).

The very first window that will appear after selecting a NAND memory from the device list is the "FRB Management", as you can see below. From this window, the user has to choose the data source which can be an existing FRB or he can quickly generate a new FRB starting from a raw binary file, an Intel HEX file or a Motorola SREC file (other options are typically not needed for NAND memories).
In the additional options, the user can choose if he wants to ignore blank pages or not. For NAND memories, this option is enabled by default and it is suggested to leave it enabled unless you know what you are actually doing because it can affect the bad block management.



Then we arrive at the NAND options, which are the most interesting part, and the very first preference to set is related to the bad block management, which can be: "Skip Method", "Reserve Method" or "No Bad Blocks". Each one of them is deeply described in the chapter "Bad block management and memory partitions".
After that, the next setting is about the memory partitions where the user has two input methods: "Import CSV" or "Edit manually". The format of the CSV must be one of the following two:

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

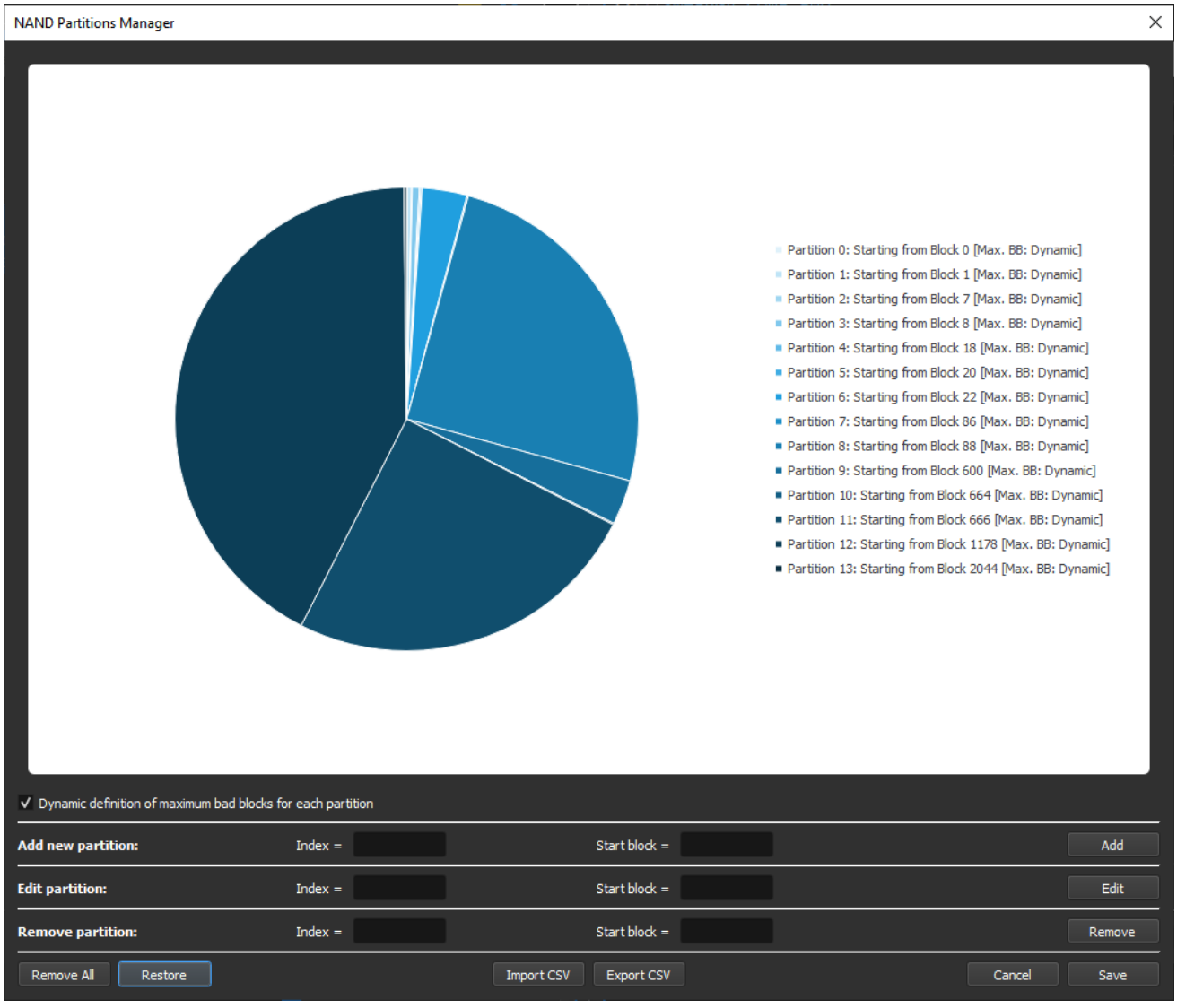
- When using the **dynamic** definition of maximum bad blocks for each partition:

```
id,startBlock
0,0
1,5
…
```

- When using the **static** definition of maximum bad blocks for each partition:

```
id,startBlock,maxBB
0,0,1
1,5,1
…
```

While, if the user chooses to manually edit the partitions, the “NAND Partitions Manager” will appear (see below) and, from this window, he will be able to do many operations.



UNIVERSAL PRODUCTION IN-SYSTEM PROGRAMMING

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

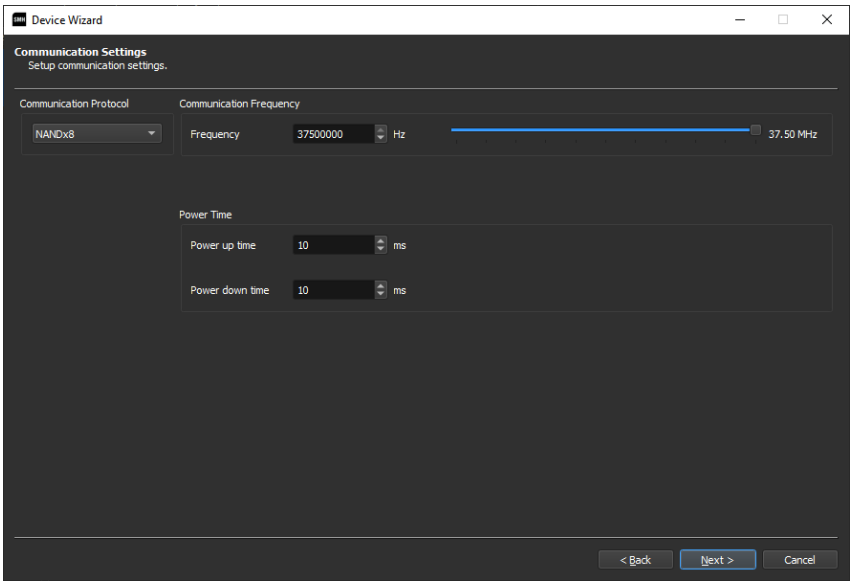
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

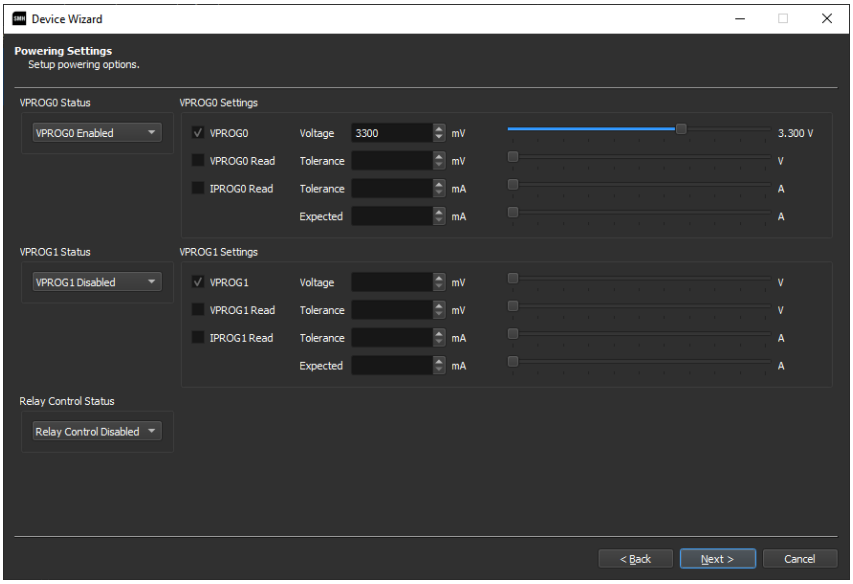
→smh-tech.com

info@smh-tech.com

The last important settings to define are related to ECC: in fact, the user must choose between “On-Die ECC” (only for those memories which are supporting it), “FRB with ECC” or “Add ECC to FRB”. When the input file is a raw binary, there will be also the possibility to remove the spare bytes from the source file (useful when using a dump file as input).
Moreover, when “On-Die ECC” is not selected, the user can also set the bit error tolerance. Anyway, all these features are deeply described in the chapter “Error Correction Code (ECC) and bit error tolerance”.

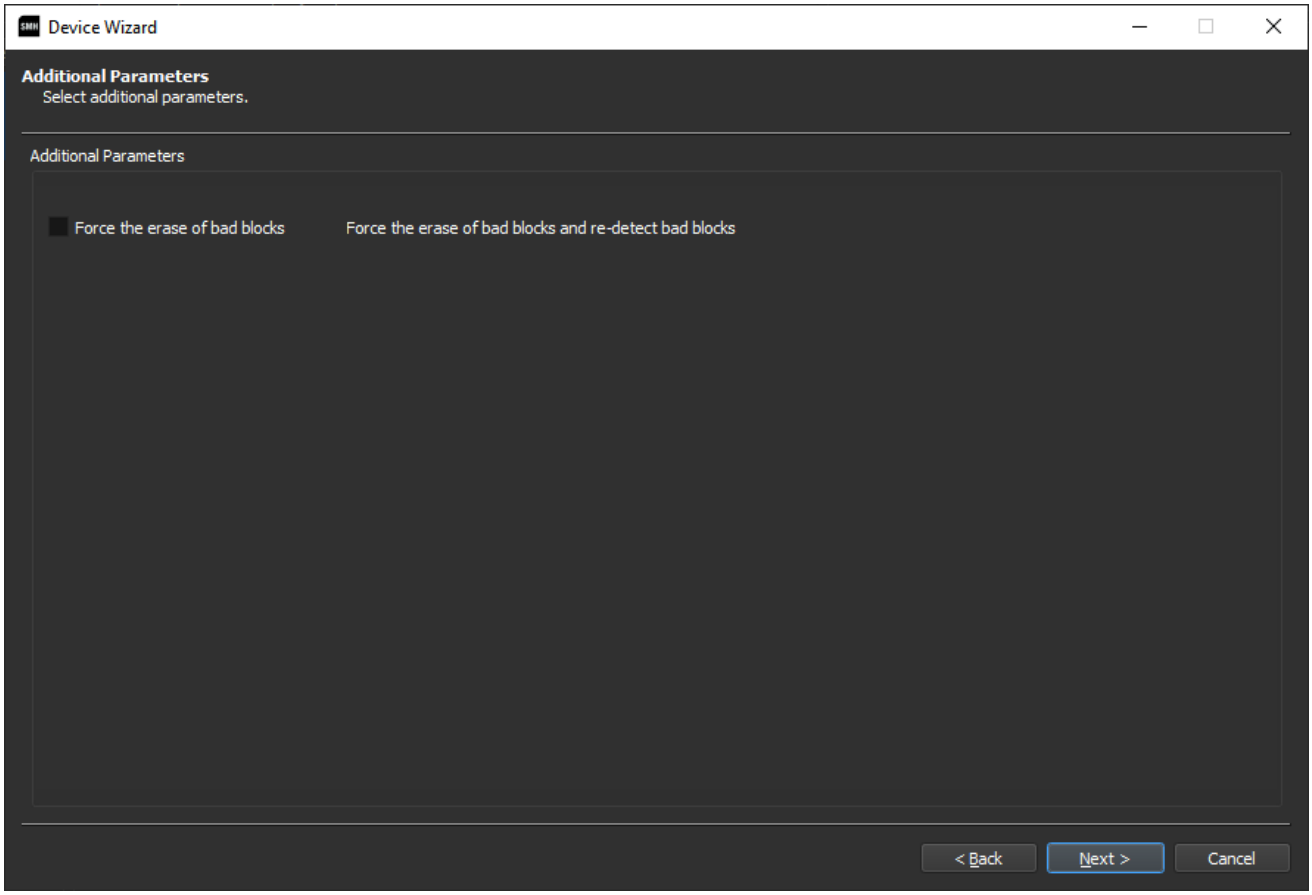
Going forward, the user has to select the communication protocol and frequency and then the voltage levels, as in the two windows reported here below.
These two windows are standard for any other device supported on FlashRunner.





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

On the next page of the wizard, the user can see the additional parameters. The only parameter is the one to enable the possibility to force the erase of bad blocks. This parameter specifies how the bad blocks are treated during the Connect and Masserase commands. When disabled, bad blocks are detected in Connect and ignored during the Masserase command. When is enabled, the bad blocks are ignored in Connect and the Masserase command tries to erase them. This parameter should be left disabled because it is useful only when performing the tests before starting production.



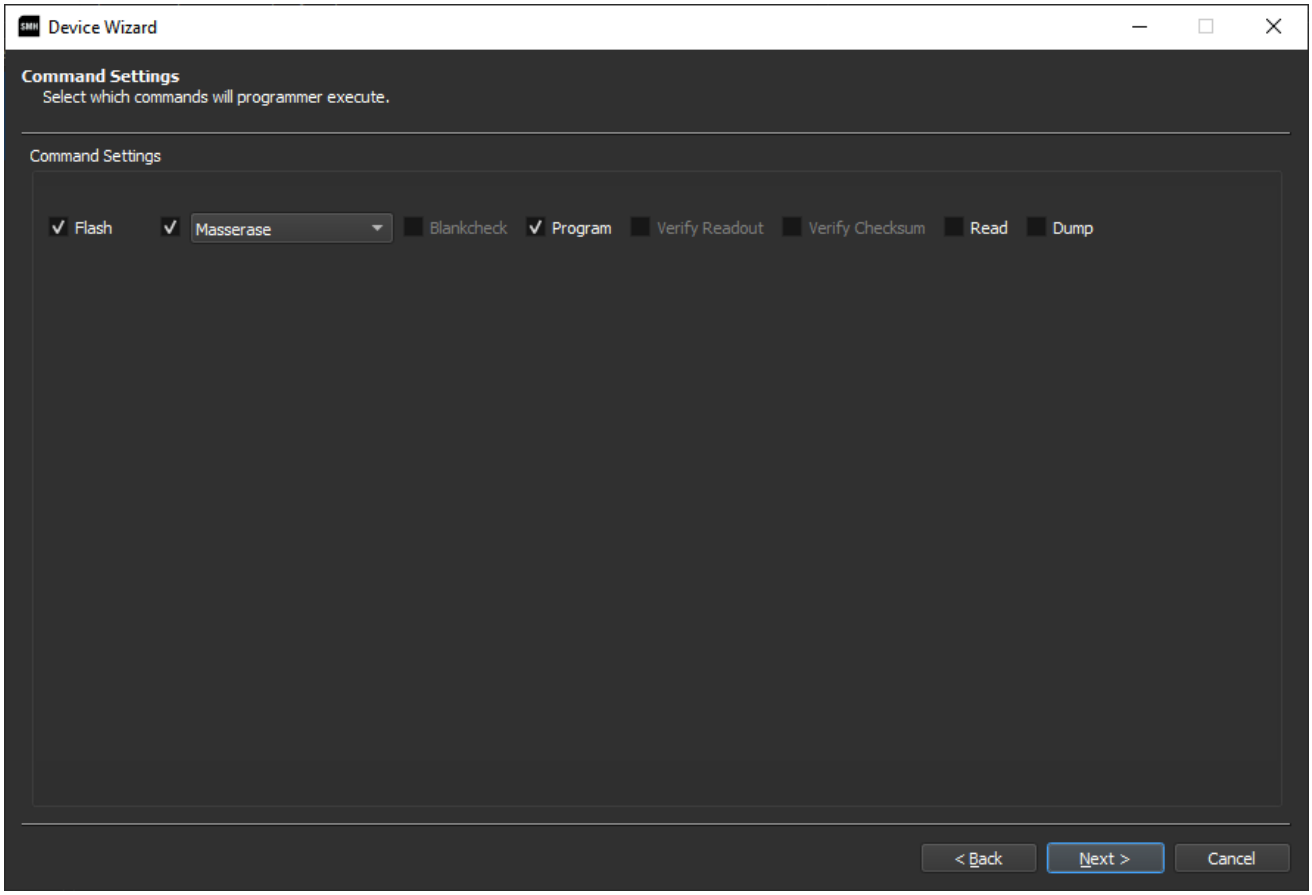
HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

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

After that, the user can choose the operations to perform on the memory. The operations suggested are the masserase, the program and, optionally, the read and dump as you can see below.



The BLANKCHECK command is not enabled by default because it is not strictly needed since the MASSERASE command is executed for each memory block and, for each one, the result of the operations is checked.
Also, the VERIFY command is not enabled by default because it is not strictly needed. In fact, the PROGRAM command is programming data block by block and it already executes the verification for each block before going forward. So, the PROGRAM command is basically executing both the PROGRAM and VERIFY commands.

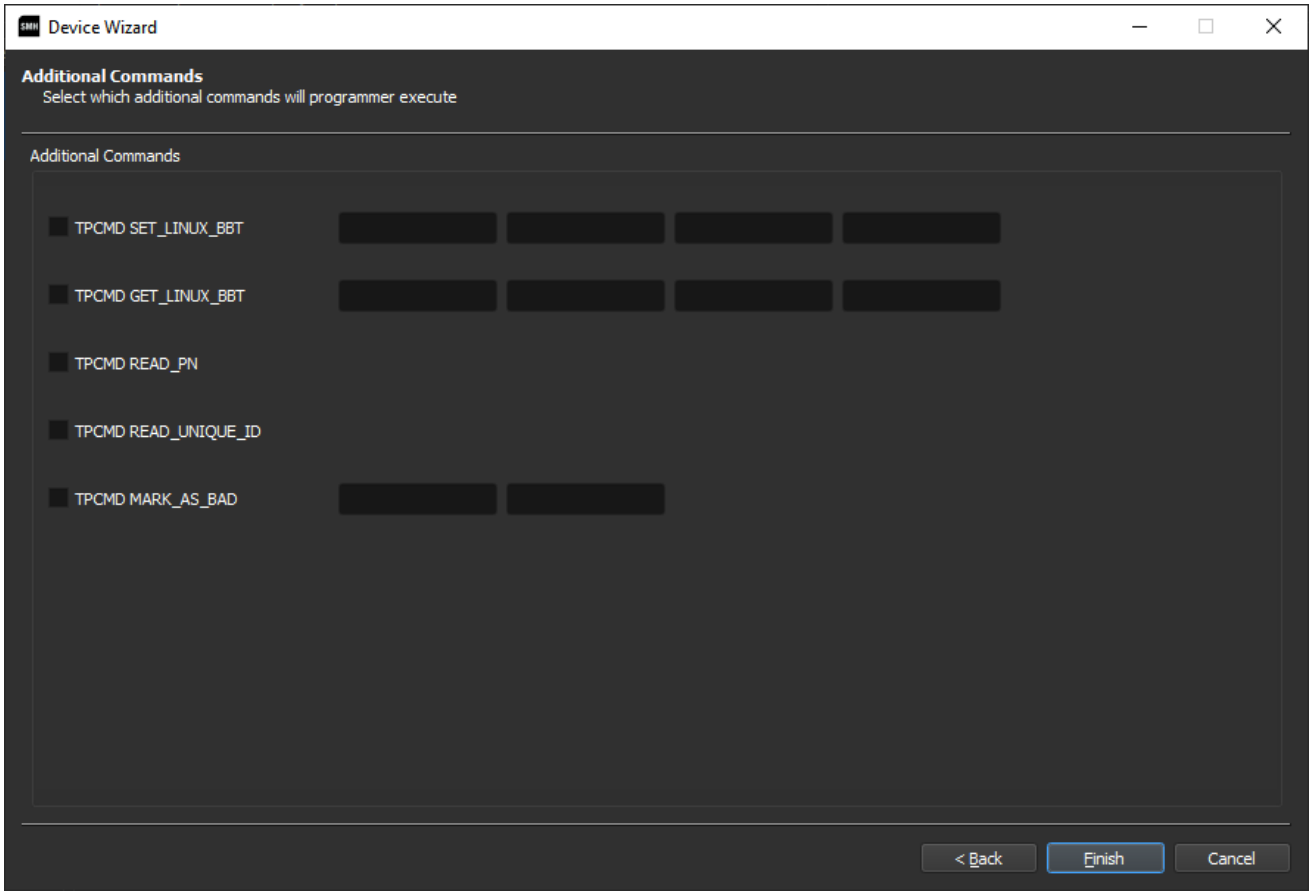
HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

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

On the last page of the wizard, the user can add additional commands to the project as shown in the screenshot below:



The additional commands available are:

- **#TPCMD SET_LINUX_BBT**
  This command composes, writes and verifies the Bad Block Table (BBT) in memory. The input values are:
  `<max_blocks (default = 4)> <pattern_offset (default = 4)> <version_offset (default = 20)> <version (default = 1)>`

- **#TPCMD GET_LINUX_BBT**
  This command reads, checks the integrity and decodes the BBT from memory. The input values are the same as in the previous command.

- **#TPCMD READ_PN**
  This command returns the part number of the connected memory. It can be used to detect the correct project to run when alternative part numbers are used. Note: this data is also present in the log during the CONNECT command.

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 READ_UNIQUE_ID**
  This command returns the UNIQUE ID of the connected memory. Note: this data is also present in the log during the CONNECT command.

- **#TPCMD MARK_AS_BAD**
  This command takes as input the block index (or a range of indices) to be marked as bad by writing all bits to 0. This command is for debugging purposes only.

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. Bad block management and memory partitions

The most important peculiarity of NAND flash memories is the presence of blocks of memories that are not properly usable and these blocks are called “bad blocks”. So, when programming data into a NAND memory, it is possible that some data defined in the customer’s firmware is defined to be inserted into a block that is actually bad, then what should we do with that? Actually, there are several possibilities, maybe infinite, because it is possible to customize the bad block management in various ways. Anyway, these are the methods supported by FlashRunner:

- **Skip Method**
  This is the most used method, especially for Linux-based systems. When this method is applied, the data that should be programmed into a bad block are shifted into the next good block. So, every time that a bad block is encountered, data are shifted forward.
  This mechanism is reset every time a new partition starts because every partition must start at a fixed address. It is also important that a partition does not overflow into the next partition when encountering bad blocks. In fact, FlashRunner can detect the overflow condition when too many bad blocks are encountered and it stops the flashing operations and returns the error “**`Partition XX overflow (too many bad blocks)`**”.

- **Reserve Method**
  This method is not meant to be used in production, this is mainly a demonstration of a different application. Using this method the customer can define a “Reserved area” which contains the data for each bad block encountered while programming the rest of the memory. The user can define the start address and size of the reserved area, so it can be combined with different partitions and different FRB files multiple times in the same project.

- **No bad blocks**
  This is a special kind of management that basically does not allow the presence of any bad blocks. When a bad block is encountered, FlashRunner will just stop the flashing operations and return an error.

If your application requires a different and customized bad block management that is not supported, feel free to ask for a quotation to support it. We are very open to support custom features since FlashRunner is a very flexible product.
Some new serial NAND flash memories from Winbond and Macronix are also offering the possibility to manage the bad block with an internal lookup table but, currently, this feature is not implemented yet. So, if you need this feature, you need to send a request to Sales@SMH-Tech.com.

As mentioned above, memory partitions play an important role, especially when applying the skip method. That is why the user is forced to define the index of the starting block for each one to proceed with the flashing operations. Optionally, it is possible to statically define the maximum number of bad blocks for each partition and, when they are defined, the flashing operations will be terminated during the connect command if too many bad blocks are detected on some partitions.

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

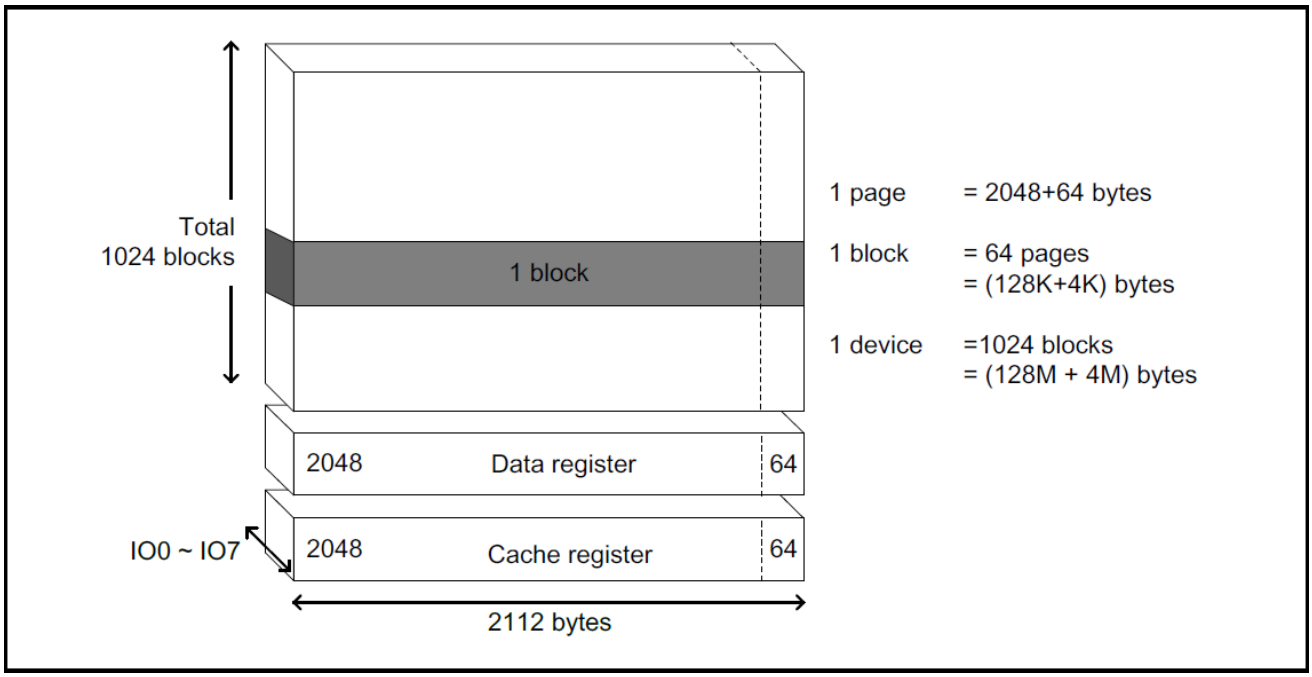
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

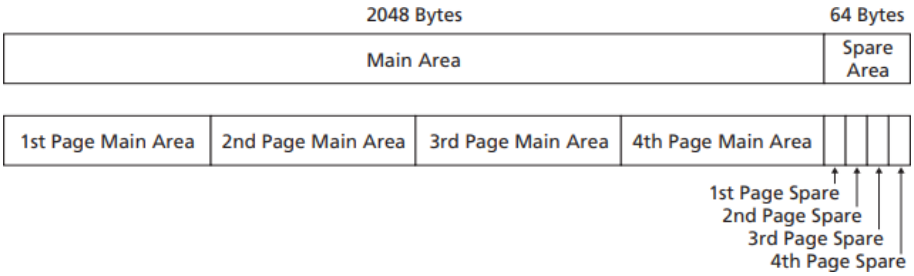
5. Error Correction Code (ECC) and bit error tolerance

The second most important peculiarity of NAND flash memories is that they can have bit-flips, so the user may write some data but the data that are actually written in the memory (or that are read back from the memory) could be different. That is why it is fundamental to have an ECC algorithm that can correct these wrong data.

Before going deep into this topic, let us give an overview of the memory structure. NAND memories are composed of blocks, each block is composed of pages and each page is divided into the data area and spare area, also known as spare bytes or out-of-band (OOB) area. Below you can see an example taken from the datasheet of a parallel NAND memory of Winbond.



Typical dimensions of pages are around 2048 bytes for the data area and 64 or 128 bytes for the spare area. Usually, the spare area contains the ECC for the data area, as you can see in the example below taken from Micron's Technical Note TN-29-63, where the data area is divided into four 512-bytes chunks and, for each chunk, there is a dedicated portion of the spare area containing its ECC.



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 ECC algorithm can be implemented in two ways: on the processor which is using the memory or directly on the memory (On-Die ECC). According to the requirements of the customer's application, the user has to choose between including ECC on the firmware when not using On-Die ECC or not including ECC on the firmware when using On-Die ECC.
In case the user needs to add ECC to the firmware and he cannot receive the full firmware from its R&D department, we can offer the possibility to add the ECC using our software tools, but this must be specified when requesting a quotation.
Another important thing to clarify when requesting a quotation is if it is needed to program dynamic data, such as serial numbers or bad block tables, and the On-Die ECC is not used, because it this case we must implement the ECC algorithm on the FlashRunner side to calculate ECC according to the dynamic data. This customization will probably result in an additional cost.

When not using On-Die ECC, FlashRunner cannot rely on the memory to automatically check the integrity of the written data and it cannot even calculate ECC because it would dramatically slow down the programming speed. The solution is to apply an error tolerance that can be defined by the user or left to the default value which is defined by the memory ("Autodetected"). In fact, the memories have a table inside which contains the "required correctable bits" and that value is used as default.

The error bit tolerance is actually an approximation because it does not take into consideration the ECC layout which could be customized in several ways. What FlashRunner does is just counting the bit errors on for each 512-bytes chunck of memory and its part of spare area and, if the error count is higher than the threshold tolerance, it will try again to refresh that page. Then, if after the refreshment, it still faces problems, FlashRunner will mark the block as bad.

One additional note about ECC that could be interesting is that, according to the algorithm used, it is possible that a blank page is different from an erased page. This happens when the ECC for a stream of 0xFF bytes is not 0xFF as well, for example when using BCH algorithm. So, it is very important to properly use the "ignore blank page" feature according to the firmware you are using.

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. Protocols and frequencies supported

In this document, we are treating parallel NAND and serial NAND in the same way, but when speaking about physical protocols, they are very different.

Serial NAND memories use basically the same physical protocol as the serial NOR memories, so you can find complete info on the document for Serial NOR flash memories and we are not going to duplicate info here.

While, regarding parallel NAND memories, the story is pretty different and the protocol used is quite complex because a lot of lines are used. The parallel NAND protocol can be 8-bit or 16-bit, where the number of bits corresponds to the number of data lines used to transfer data. Currently, we are only supporting the 8-bit bus because the 16-bit protocol is deprecated but, if you really need it, you can always ask for a quote.

The lines used by the parallel NAND protocol are the ones reported below:

FlashRunner	NAND
DIO0	CLE
DIO1	ALE
DIO2	CE0#
DIO3	WE#
DIO4	RE#
DIO5	CE1# (If present)
DIO6	CE2# (If present)
DIO7	CE3# (If present)
DIO8	IO0
DIO9	IO1
DIO10	IO2
DIO11	IO3
DIO12	IO4
DIO13	IO5
DIO14	IO6
DIO15	IO7

For both serial and parallel NAND, the maximum frequency supported is 37.5MHz and it directly influences the cycle time because it determines the data rate.

Attention: if you notice some instabilities, 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

7. 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 performances. In fact, the quality of the connections between the FlashRunner and the target board is extremely important and in this chapter we want to define some guidelines.

When using parallel NANDs or serial NANDs in Quad-SPI or Octo-SPI, you have a lot of data lines changing states at the same time and the quality of the wirings is crucial:

- Use our flat cables with cable interface to go as near as possible to the target device.
- The wiring should be as short as possible and all the wiring should have all the same length.
- There should be as few discontinuities as possible (i.e. prefer one 30 cm cable instead of two 15 cm cables connected together).
- Add more ground wires between FlashRunner and the target device, possibly twisted with each signal line, especially with the clock signal (also called RE# on parallel NAND memories).

In fact, these protocols are not designed to be robust to be used for In-System programming. They were designed to guarantee a high data rate on the communication bus between the memory and the processor, but the memory and the processor are on the same PCB, very close to each other and using carefully designed routes.

Another common issue that can arise is the crosstalk: the phenomenon by which signals transmitted on one or more lines generate undesired effects on other lines due to electromagnetic coupling among them. For example, when transmitting 0x00 followed by 0xFF in Octo-SPI, the simultaneous switching of eight data lines from 0 to 1 can pull the clock line to 1 and generate a spurious clock pulse.
To mitigate this effect, one possible solution is to add small resistors (i.e. 100 Ohm for 3.3V devices or 56 Ohm for 1.8V devices) in series on all data lines. These resistors decrease the current flow on data lines and then reduce the interaction effect on the clock line.
One more useful trick to protect the clock line from the crosstalk is to add a small capactitor (i.e. about 20~30 pF) between the clock and the ground line on the fixture side.

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

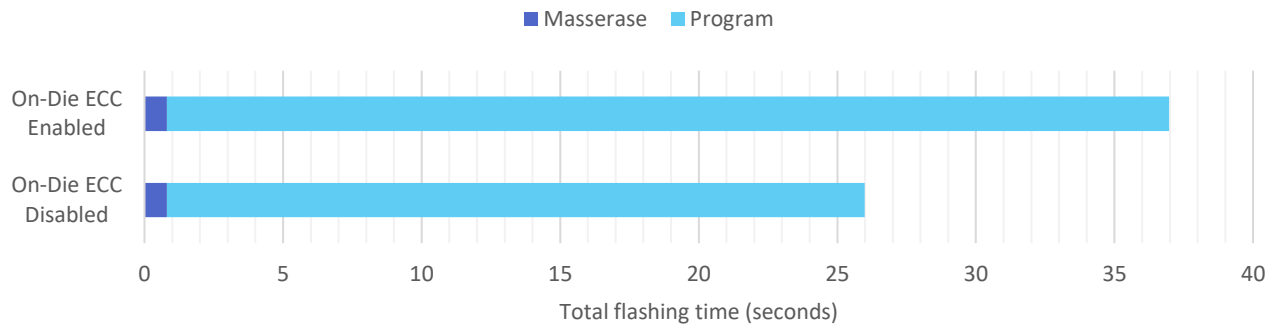
D-U-N-S® 51-724-9350
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

8. Flashing time examples

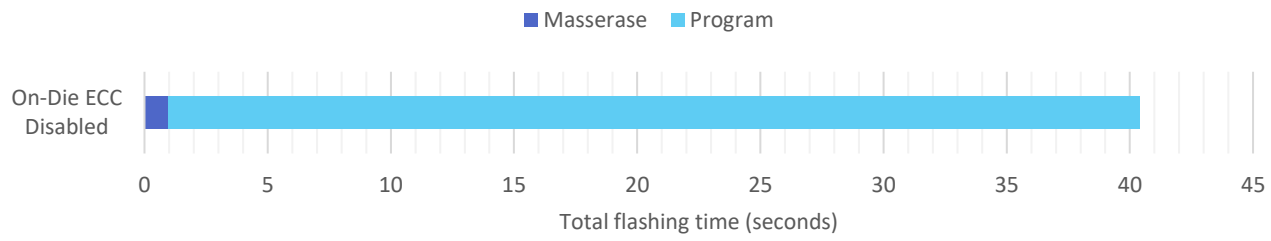
These are some examples of flashing times using our new drivers >= 2.03 and OS >= 3.19. Times were measured using FRB files containing random data to cover the 75% memory of the target device to simulate an actual customer firmware that is never filling the full memory with real data.

MT29F2G08ABAEA [2 Gbit]



Protocol	Firmware size	Masserase	Program (and verify)
NANDx8 37.5 MHz	192 MB (Without ECC) On-Die ECC enabled	0.81 s	36.16 s
NANDx8 37.5 MHz	198 MB (With ECC) On-Die ECC disabled	0.81 s	25.18 s

W29N02GV [2 Gbit]



Protocol	Firmware size	Masserase	Program (and verify)
NANDx8 37.5 MHz	198 MB (With ECC) On-Die ECC disabled	0.95 s	39.45 s

Note: this memory does not have the On-Die ECC available.

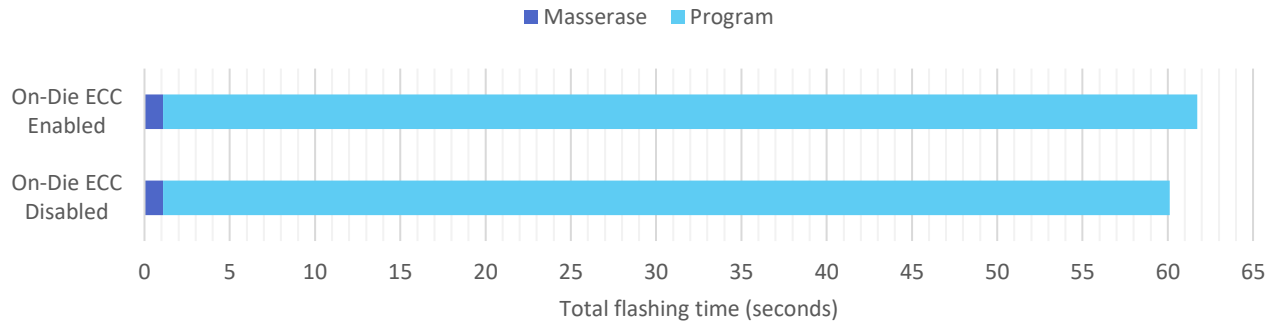
HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

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

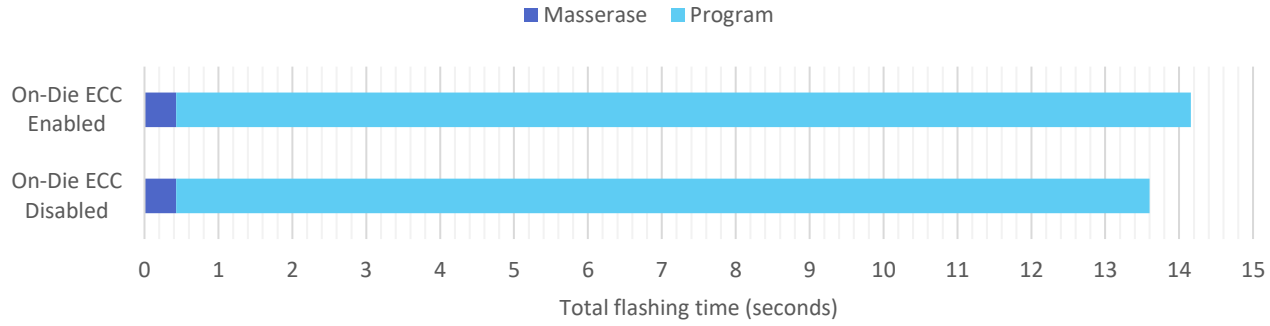
→smh-tech.com
info@smh-tech.com

W25N02JW [2 Gbit]



Protocol	Firmware size	Masserase	Program (and verify)
Quad-SPI 37.5 MHz	192 MB (Without ECC) On-Die ECC enabled	1.09 s	60.64 s
Quad-SPI 37.5 MHz	198 MB (With ECC) On-Die ECC disabled	1.09 s	59.02 s

W35N01JW [1 Gbit]



Protocol	Firmware size	Masserase	Program (and verify)
Octo-SPI 37.5 MHz	96 MB (Without ECC) On-Die ECC enabled	0.43 s	13.73 s
Octo-SPI 37.5 MHz	99 MB (With ECC) On-Die ECC disabled	0.43 s	13.17 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

9. Frequently Asked Questions

Some answers to the most frequently asked questions:

How many channels are needed to use a NAND memory?

The answer to this question is not so easy because it depends on the protocol used.
For SPI or Quad-SPI memories, it is easy because only one channel is used and they are treated as any usual device.
On the other hand, for parallel NAND memories using 8-bit bus or for Octo-SPI memories, more ISP lines are needed and they cannot be managed by a single channel, so a pair of channels is needed, for example, channels 1 and 2, or 3 and 4, etc.
This implicates some additional limitations on the *FlashRunner NXG* and the *FlashRunner 2.0*: when using this special mode, the FlashRunner can parallelly flash only memories which employ the same protocol. It is not possible to flash devices which is not a parallel NAND when using the FlashRunner in 8-bit mode. For example, in case the user has to flash an Aurix TC397 using the same FlashRunner used to flash a parallel NAND, then the Aurix TC397 can be connected to an empty channel and it can be flashed before or after performing the operations on the NAND memory, but not at the same time. In these situations, typically customers prefer to use two independent FlashRunner to use one in 8-bit mode and the other one in standard mode.
Instead, for the *FlashRunner HS*, the situation is easier because there are dedicated active modules that do the job, so more combinations are possible. *FlashRunner HS* is generally suggested for NAND memories because it can be integrated with active modules very close to the target, this means better signal integrity and, so, the possibility to reach a higher data rate without compromising stability.

How bad blocks are detected?

Bad blocks can be detected during different flashing operations:

1. During the connect command, FlashRunner scans the entire memory reading the first byte of the spare area of the first page of each block. If that byte is different from 0xFF, then that block is recognized as bad.
2. During the masserase command, FlashRunner executes the erase command for each block and checks the result of that operation. If the erase operation fails, then that block is recognized as bad.
3. When programming data, FlashRunner checks the result of the write operation and, if it fails, then the block is recognized as bad.
4. When verifying data after programming, if there are uncorrectable errors by ECC or mismatches in the data, FlashRunner tries to refresh the data on the block. If there are still problems after the refreshment, the block is marked as bad.

When FlashRunner detects a bad block, it marks all the bytes of all the pages of that block as 0x00.

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

SMH Technologies®

How does the dynamic detection of maximum bad blocks work?

When applying the skip method for bad block management, FlashRunner can detect when the maximum amount of bad blocks is reached by a partition because it checks that there is not a partition overflow. To make it simple, consider a partition that has 10 blocks in total and 9 blocks of data. When finding a bad block in the middle of that partition, the last block of data will be programmed in the last available block. When finding 2 bad blocks in the middle of the partition, the last block of data should be shifted forward, overflowing the partition. In this last case, FlashRunne notices the problem and returns an error interrupting the flashing operation.

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