SMH Technologies®

DC11545
Driver 2.21
July 2023
R. Ertolupi

Interfacing FlashRunner with Power PC MCUs [56] (MPC56xx and SPC56xx)

1. Introduction

The Power PC MCUs described in this document are the ancestors of the MPC57xx and SPC58xx: their story and their applications are the same. In fact, there are a lot of common characteristics and that is why you will notice many similarities to the documentation for MPC57xx and SPC58xx. Let us consider this document for MPC56xx and SPC56xx an extension of the one for MCP57xx and SPC58xx that can be found here: Interfacing FlashRunner with Power PC MCUs.

You can download the latest version of this document from this static link: Interfacing FlashRunner with Power PC MCUs [56].

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. Tips and tricks to flash Power PC MCUs

In this section, we want to explain what operations are performed in the target devices and how these operations work. This knowledge may help you to optimize even more your application.

The FlashRunner uses JTAG to access to the memories of these devices, which is one of the most common protocols for microcontrollers and which needs four synchronous lines (TCK, TMS, TDI, and TDO) and an optional asynchronous line (TRST). In addition to standard JTAG lines, the FlashRunner also need to control the reset line of the target device: this line is very important and very critical as well. It is crucial to have no interferences on the reset line because it could cause strange behaviours of the device and the operations could likely fail.

The new driver, combined with an OS > 3.00, can support a clock frequency for JTAG up to 37.5 MHz, which also helps to reduce flashing times. However, there are a couple of things to say about that.
The first one is that the customer needs a good hardware setup to be able to reach such a high frequency, otherwise the quality of the signals will be bad and operations could fail.
The second aspect to consider is that the JTAG frequency is proportionally limited to the system clock frequency (typically the limit is SYS_CLK /2 or /4). Exactly for this reason it is fundamental to enable the PLL and that can be done when creating a project or editing the project file. As you can see from the image below, when creating a project from our user interface, it is possible to define the frequency of the external oscillator mounted for the device and the PLL frequency of the device.

After creating the project, these parameters will be encoded with these two lines:

```
#TCSETPAR FOSC 30000000
#TCSETPAR FPLL 150000000
```

For some devices the input clock is fixed to a single value because the internal oscillator is used, so there are no choices for the user. So, you may wonder why we do not fix the PLL frequency as well to simplify the usage, and the answer is: because most of these devices are available in different revisions which differ for the maximum operating frequency supported, so the user must pay attention to the actual part number used as target device and choose the correct PLL frequency.
For some devices, especially the ones which are less powerful, even by enabling the PLL, they cannot support 37.5 MHz as JTAG clock but the user should not worry about that because the driver is checking if the required specifications are respected and it automatically will lower the JTAG frequency if needed.

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

Please, note that the configuration of the PLL done by the FlashRunner is volatile and it will be lost after resetting the device, so it is completely transparent to the customer.

Enabling the PLL of the device not only allows to speed up the communication between the FlashRunner and the Power PC MCU, but also makes the target device much more performant when executing operations. This is even more relevant since we use the "*Standard Software Driver*" (aka microkernel), which is a set of pre-compiled functions supplied by silicon producers to operate on the device. Let us analyse how this can influence the standard commands:

- **Masserase**
  This command erases all the content of the memory selected. The supported memories for this command are: Code Flash and Data Flash.
  Optionally, it is also possible to send four additional parameters to this command to select only some sectors of memory to be erased. This can be useful to reduce the erase time and it could be needed if the user wants to preserve some data into the memory and partially reprogram it. See the chapter "Sector Erase" for more information.
  The duration of Masserase depends only on the target device characteristics and there is not much that we can do to improve this. Typically, it is extremely long, even longer than all the other operations combined together.
  The suggestion is to skip this operation in case the device is virgin. This can be done using a conditional script based on the result of the blankcheck operation:

```
#IFERR TPCMD BLANKCHECK F
#THEN TPCMD MASSERASE F
#THEN TPCMD BLANKCHECK F
```

- **BlankCheck**
  This command checks that all the bits of the selected memory are set to 1. The supported memories for this command are: Code Flash and Data Flash. This operation is executed internally by the device and it is typically extremely fast.
  Optionally, it is also possible to send two additional parameters to this command: the address from where to start checking and the number of bytes to check.

  The duration of this operation depends on the size of the memory and by the internal frequency of the Power PC MCU. So, enabling PLL will reduce a lot the duration of this command and it can be about ten times faster.

- **Program**
  This command takes the customer's data from the FRB file and programs them into the selected memory, the supported memories for this command are: Code Flash, Data Flash, Shadow and OTP Test 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.

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 duration of this command depends on many factors which are all important:

- The target device characteristics, in other words, how much time is needed to write data into the memory.
- The target device frequency because it determines how fast the device is to process data, so it is important to have the PLL enabled.
- The JTAG frequency because it sets the bitrate of the communication between FlashRunner and the target device.

The suggestion to improve the performances of this command is to set the FRB file with the "IGNORE_BLANK_PAGE" option. This will skip the program operation for any page which contains only 0xFF bytes.

```
#TPSETSRC myData.frb IGNORE_BLANK_PAGE
```

- **Verify Readout**

  This command checks that data contained in the memory of the device corresponds to FRB data. The supported memories for this command are: Code Flash, Data Flash, Shadow and OTP Test Memory.
  Optionally, it is also possible to send two additional parameters to this command: the address from where to start checking and the number of bytes to check.

```
#TPCMD VERIFY F R 0xFC0000 0x1000
```

  This command works exactly like the program command with the only exception that the device reads (instead of writing) data from the memory and compares that with the data received from FlashRunner.
  Since the mechanism is totally equal to the one used by the program command, if any error was introduced during the program command, it is possible that the same error could be introduced during the verify command and this could lead to a possible undetected error. For this reason, we suggest using the verify checksum instead of the verify readout, or maybe combining them.

- **Verify Checksum**

  This command asks the target device to calculate the 32-bit checksum of the selected memory region, meanwhile, the FlashRunner calculates the expected checksum according to FRB data and then the two values are compared. The supported memories for this command are: Code Flash, Data Flash, Shadow and OTP Test Memory. This operation is executed internally by the device and it is typically extremely fast, almost like the blankcheck.

```
#TPCMD VERIFY F S
```

  Optionally, it is also possible to send up to three additional parameters to this command: the address from where to start the checksum calculation, the number of bytes to consider in the

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

calculation, and the expected checksum value. This will result faster because the FlashRunner does not need to spend time calculating the checksum of the FRB file.

```
#TPCMD VERIFY F S 0x00000000 0x00001000 0x4610E323
```

To get the value to use as expected checksum parameter, you can use the command below, which can be executed by FlashRunner without being connected to the target device because it is just an internal calculation and it will return the commands to use in the real-time log.

```
#TPCMD CALC_FRB_CHKSUM
```

- **Margin check**
  This optional command can be used by customers to check that the voltage levels of all the memory cells are at the proper level. This is useful to guarantee the expected data retention. Basically, this command asks the device to calculate the MISR values for each bank of memory and for both threshold levels: high and low. Then the driver will check that MISR values obtained using high threshold correspond to the ones obtained using low threshold. Attention: this command is not verifying that the memory content is matching with the FRB data, it is just checking the memory status, so users should call this command after the verify command. This margin check command can simply be called without any parameters:

  ```
  #TPCMD MARGIN_CHECK
  ```

  Warning from ST and NXP: doing margin reads repetitively results in degradation of the flash memory array, and shortens the expected lifetime experienced at normal read levels. For these reasons the margin read usage is allowed only in factory mode, while it is forbidden to use it inside the user application.

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.

Warning: performing the program and verify commands using an FRB which does not contain any data for the selected memory region will return pass. The driver has been designed in this way to be more flexible so, basically, if a customer gives no data to program and verify, then the driver does not perform any operation and it just returns pass after completing the research for the data.

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. Sector erase

As already said in the previous chapter, the user can also choose to erase only some sectors of memory just by setting some more parameters to the masserase command. This could be the most optimized choice in case the erase cannot be skipped.

```
#TPCMD MASSERASE <MemType> <SEL_LOW> <SEL_MID> <SEL_HIGH>
```

The parameters of this command are:

- **MemType**
  This indicates the memory chosen, it can be one of the following characters:
  - **F** – Code Flash memory.
  - **E** – Data Flash memory.

- **SEL_LOW SEL_MID SEL_HIGH**
  These are the bitmasks that select the block of memory to erase. It is forbidden to select a block that is not part of the **MemType** selected.

Usage example:

```
#TPCMD MASSERASE F 0x3F 0x0 0x0
```

Using bitmasks rather than addresses is a design choice that we made to get higher reliability. In fact, the sectors have different sizes and they are not in a fixed order for all devices so, decoding addresses to bitmasks from the software is very complex and complexity does not go along with reliability. Moreover, also the API of ST and NXP work in this way, so we are following their choice as well.

To check that the operation was successfully done, it is possible to perform the blankcheck operation specifying the address and size, so the user will be able to check that the memory sectors were properly selected. For example:

```
#TPCMD BLANKCHECK F 0x0 0x20000
```

The command to erase only some sectors of memory may not seem user-friendly at all but, with some explanations, it could be easier to understand. In fact, choosing the bits to select the sectors of memory to be erased requires a bit of effort and just by following the Reference Manual of the device.

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

5. Shadow memory: lock and unlock using JTAG password

Shadow memory is a very special memory, which should never be programmed randomly until you want to trash the device. Since this memory is so critical, in this chapter we want to explain how to properly manage it using FlashRunner.

FlashRunner does not allow the user to erase this memory without re-programming some data, so, the customer can only use the program and verify commands. When calling the program command without specifying the memory addresses, FlashRunner will automatically erase the Shadow memory before writing new data. Otherwise, if the program command is called specifying the address and the size of the data to be programmed, the Shadow will not be erased to allow only the selected data to be overwritten.

The Shadow memory contains important settings that allow the user to protect the device using the JTAG password in various ways. For these devices, the size of the JTAG password is 64-bit. To access a locked device the programmer must send the JTAG password during the execution of the connect command. The user must set the FlashRunner to use the JTAG password before calling that command and this can be done by setting this parameter to YES:

```
#TCSETPAR PASS_EN YES
```

After enabling the JTAG password, the user must send the value of the password to use to unlock the device and this can be done using dynamic memory (or FRB) at 0x00203DD8, which corresponds to the actual memory address of the password inside the Shadow memory. Warning: 0x00203DD8 is a typical address, there are some devices which have a different base address for the Shadow region, so the password should be placed at: base address + 0x3DD8.

Alternatively, it is also possible to use the command: **`PWD_ENTRY <32bit_1> <32bit_2>`**. Which sets the password to unlock the device and which must be called before the connect command. It receives 8-byte (two 32-bit words) as parameter to set the password to use.

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

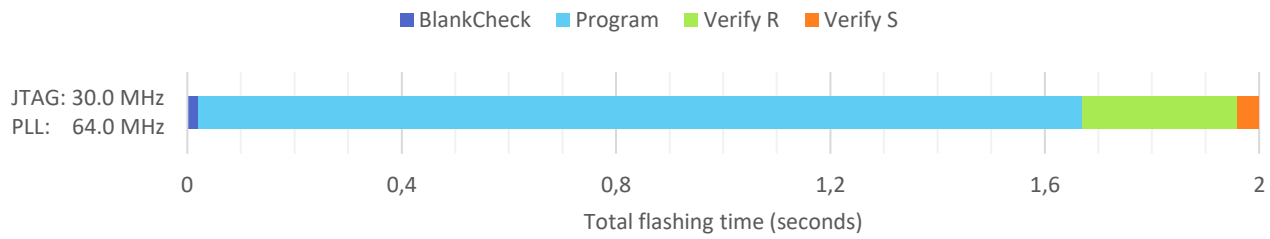
6. Flashing time examples

These are some examples of flashing times using our new drivers and OS > 3.00.

Additional notes:

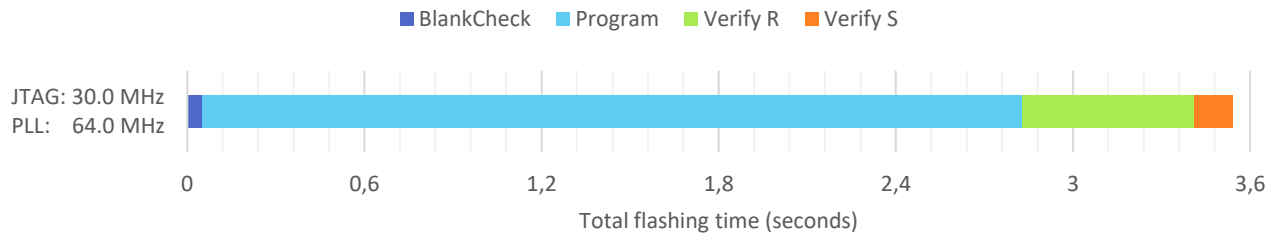
1. Times were measured using FRB files containing random data to cover the entire code flash memory of the target device.
2. We do not consider the masserase time in the total flashing time because it is usually skipped.
3. Verify checksum executed sending the pre-calculated checksum as a parameter.

MPC5604B [0.5 MB]



Frequency	Masserase	BlankCheck	Program	Verify R	Verify S
JTAG: 30.0 MHz PLL: 64.0 MHz	2.32 s	0.02 s	1.65 s	0.29 s	0.04 s

SPC56AP60 [1 MB]



Frequency	Masserase	BlankCheck	Program	Verify R	Verify S
JTAG: 30.0 MHz PLL: 64.0 MHz	4.04 s	0.05 s	2.78 s	0.58 s	0.13 s

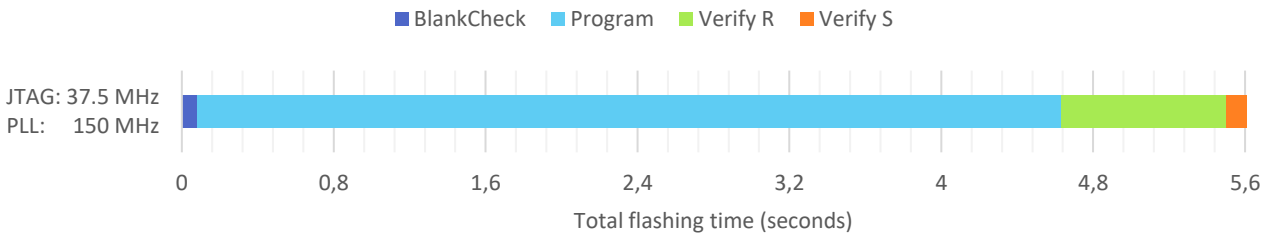
HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

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

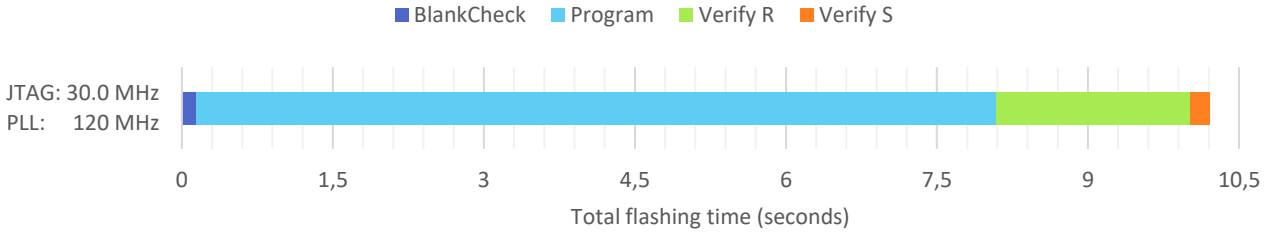
→smh-tech.com
info@smh-tech.com

SPC564A70 [2 MB]



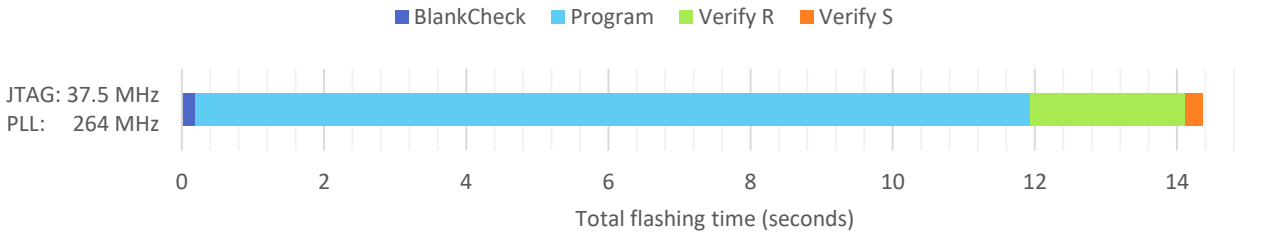
Frequency	Masserase	BlankCheck	Program	Verify R	Verify S
JTAG: 37.5 MHz PLL: 150 MHz	12.88 s	0.08 s	4.55 s	0.87 s	0.11 s

SPC564B74 [3 MB]



Frequency	Masserase	BlankCheck	Program	Verify R	Verify S
JTAG: 30.0 MHz PLL: 120 MHz	8.84 s	0.14 s	7.95 s	1.93 s	0.19 s

MPC5674F [4 MB]



Frequency	Masserase	BlankCheck	Program	Verify R	Verify S
JTAG: 37.5 MHz PLL: 264 MHz	20.53 s	0.19 s	9.86 s	1.97 s	0.25 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

Other flashing time examples

These are some flashing times of other devices. Time for masserase is not reported because it is typically skipped for virgin devices and total time is considering both verify readout and checksum.

Device	Size [PLL Freq.] [JTAG Freq.]	BlankCheck	Program	Verify R	Verify S	Total
SPC560D40	0.25 MB [48.0 MHz] [20.0 MHz]	0.03 s	4.68 s	3.75 s	0.04 s	8.50 s
MPC5602P	0.25 MB [64.0 MHz] [30.0 MHz]	0.05 s	3.29 s	2.29 s	0.01 s	5.73 s
SPC560P40	0.25 MB [64.0 MHz] [30.0 MHz]	0.01 s	3.58 s	2.76 s	0.02 s	6.37 s
MPC5604E	0.5 MB [60.0 MHz] [30.0 MHz]	0.01 s	1.62 s	0.29 s	0.07 s	1.99 s
SPC560B50	0.5 MB [64.0 MHz] [30.0 MHz]	0.02 s	1.74 s	0.29 s	0.03 s	2.08 s
MPC5606B	1 MB [64.0 MHz] [30.0 MHz]	0.04 s	2.59 s	0.58 s	0.07 s	3.28 s
SPC560P60	1 MB [64.0 MHz] [30.0 MHz]	0.05 s	2.80 s	0.80 s	0.13 s	3.78 s
SPC564L60	1 MB [120 MHz] [30.0 MHz]	0.05 s	3.02 s	0.58 s	0.07 s	3.72 s
MPC5606S	1 MB [64.0 MHz] [30.0 MHz]	0.04 s	3.39 s	0.59 s	0.04 s	4.06 s
SPC560B64	1.5 MB [64.0 MHz] [30.0 MHz]	0.06 s	4.07 s	0.87 s	0.10 s	5.10 s
SPC56EC64	1.5 MB [120 MHz] [30.0 MHz]	0.05 s	4.10 s	0.86 s	0.09 s	5.10 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

SPC563M64	1.5 MB [64.0 MHz] [30.0 MHz]	0.07 s	5.15 s	0.86 s	0.15 s	6.23 s
MPC5645S	2 MB [124 MHz] [30.0MHz]	0.09 s	4.86 s	1.07 s	0.14 s	6.16 s
SPC56EL70	2 MB [120 MHz] [30.0MHz]	0.10 s	7.37 s	1.39 s	0.14 s	9.00 s
MPC5646C	3 MB [120 MHz] [30.0 MHz]	0.17 s	8.23 s	1.97 s	0.20 s	10.57 s
MPC5644A	4 MB [150 MHz] [37.5 MHz]	0.45 s	11.19 s	2.12 s	0.58 s	14.34 s
MPC5676R	6 MB [180 MHz] [37.5 MHz]	0.41 s	9.82 s	2.60 s	0.60 s	13.43 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

7. Frequently Asked Questions

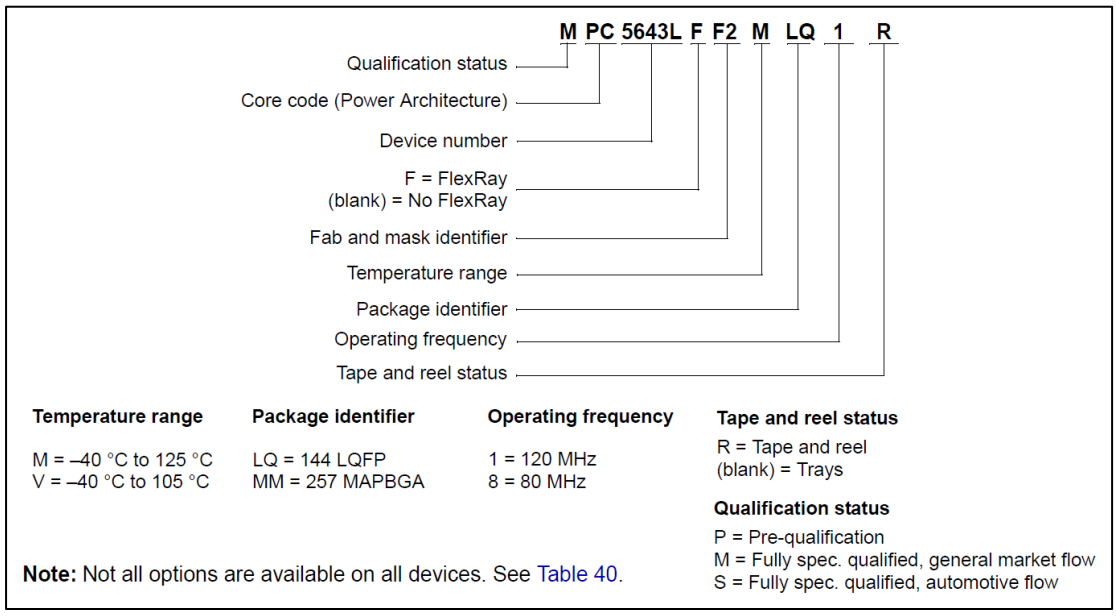
Some answers to the most frequently asked questions:

Are these new updates available for all FlashRunner models?

No, these updates are only for the *FlashRunner NXG*, the *FlashRunner 2.0*, and the *FlashRunner HS*. These newer models are more powerful and they can be much faster than the FlashRunner of the classic series.

What are the differences between MPC56xx, PPC56xx, and SPC56xx?

From our point of view, there are no significant differences. NXP (Freescale) uses this particular standard for their Power PC MCUs, where the first letter indicates the qualification, as you can see in the example below. Since we just do in-system programming, for us they are equals and that letter is irrelevant. That is why in our device list all three versions with different qualifications are merged and named as MPC56xx.



What about MPC57xx and SPC58xx?

We are also supporting them on a different driver because they have different characteristics. Find more details on their dedicated document: Interfacing FlashRunner with Power PC MCUs.

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

What about MPC55xx?

We are also supporting them on a different driver PowerMCU55 because they have different characteristics. They are deprecated devices and they have limited features, so we are just very basic operations without any lock-unlock option.

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