



02/02/2026

Driver v. 1.00

Antonio Di Bello

Marco Colombi

Ivan Liberotti

Interfacing FlashRunner HS with MIMX8 MM/MN USB Flashing



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

USB Protocol

MIMX8 MM/MN microprocessors could be used to program huge size memories through its USB interface.

FlashRunner HS combined with the USB active module allows the user to connect the USB interface of the microprocessors to SMH programming system and then send data and USB commands to program the huge size memories connected to them.

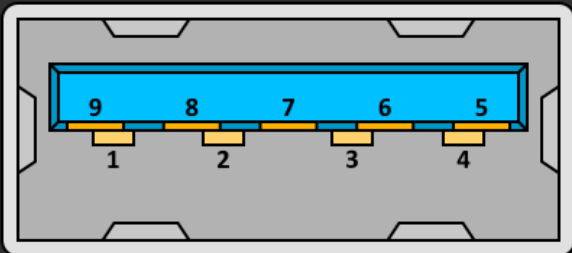
#TCSETPAR CMODE <USB>

USB Pin Map

Pin Map Tool

Select your FlashRunner model: **FR HS** Export to PDF

Active Module 1 - Connector 1



Select a channel:

- A.M.1 Ch.1 - MIMX8MM6CVTKZA_EMMC16G-WW28 [USB]

Connection descriptions:

| | |
|------------|--------|
| VBUS (5V) | Pin: 1 |
| D- | Pin: 2 |
| D+ | Pin: 3 |
| GND | Pin: 4 |
| STDA_SSRX- | Pin: 5 |
| STDA_SSRX+ | Pin: 6 |
| GND | Pin: 7 |
| STDA_SSTX- | Pin: 8 |
| STDA_SSTX+ | Pin: 9 |

Sparse files Handling

For this kind of algorithm, it is necessary to develop a way to handle sparse files which are commonly used to program huge size memories through USB protocol, in particular when using Fastboot communication.

For this purpose, SMH developed its own command line utility to handle the conversion of a sparse file into a raw binary file

```
C:\> C:\Windows\System32\cmd.exe
```

```
Microsoft Windows [Versione 10.0.19045.4894]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\iliberotti\Desktop\UUU>imgSimgManager file.sparse file.bin -sr 512
Invalid sparse file format at header magic
Failed to read sparse file

C:\Users\iliberotti\Desktop\UUU>imgSimgManager file.sparse file.bin -sr 512

C:\Users\iliberotti\Desktop\UUU>
```

In the first part the input was wrong. In the second part you can find the command “imgSimgManager file.sparse file.bin -sr 512” and the parameters:

- 1) Name of the tool
- 2) Name of the input file
- 3) Name of the output file
- 4) Conversion flag (-sr sparse to raw, -rs raw to sparse)
- 5) Sparse file block size (optional, 512 by default)

Once this process has been completed for every sparse file, it is possible to create the usual FRB file for every binary file created. You will just need to go through the advanced FRB manager of the GUI, select the huge size memory, upload the raw binary and generate the FRB.

Programming flow

The first part consists of loading an embedded application (uboot) into the MIMX8 MM/MN RAM through a USB HID class protocol like SDP or SDPS. These protocols allow to access a USB device without OS specific drivers. The embedded application is provided by NXP.

Once this bootloader has been successfully loaded, it is possible to switch to SDPU/SDPV, which are simplified ROM SDP protocol implemented into uboot, or directly to fastboot communication and execute set of uboot and fastboot commands.

Commands

SDP/SDPS/SDPU/SDPV commands

#TPCMD SDP <argument>

Argument can be:

- 1) boot: used to program uboot through SDP command (proper source firmware file required)
- 2) dcd: used to write dcd
- 3) write: used to program IVT/boot
- 4) jump: used to jump to execution address

#TPCMD SDPU <argument>

Argument can be:

- 1) dcd: used to write dcd
- 2) write: used to program IVT/boot
- 3) jump: used to jump to execution address

#TPCMD SDPV <argument>

Argument can be:

- 1) dcd: used to write dcd
- 2) write: used to program IVT/boot
- 3) jump: used to jump to execution address

#TPCMD SDPS <argument>

Argument can be:

- 1) boot: used to program uboot through SDPS command (proper source firmware file required)

Fastboot commands

#TPCMD FB_UCMD <string>

String can be any type of string command supported by the uboot embedded application running into the device RAM. This command is used to perform uboot commands.

#TPCMD FB_ERASE <partitionName> <startAddress> [size]

This command is used to erase an eMMC partition:

- 1) <partitionName> indicates the partition targeted by the operation
- 2) <startAddress> indicated the address from which the operation should start
- 3) [size] is optional and indicates the portion in bytes to be erased

#TPCMD FB_FLASH <partitionName>

This command is used to flash an eMMC partition:

- 1) <partitionName> indicates the partition targeted by the operation

(proper source firmware file required)

IMX8_USB Standard Commands

#TPCMD CONNECT

Used to connect to the target.

#TPCMD DISCONNECT

Used to disconnect to the target.

IMX8_USB Additional Commands

#TPCMD WAIT_SWITCH_SDPV <timeout>

Used when SDP to SDPU protocol switch waiting is needed. Timeout in [ms] is the maximum wait time; after that if no switch has happened, the operation will fail.

#TPCMD WAIT_SWITCH_SDPV <timeout>

Used when SDP to SDPV protocol switch waiting is needed. Timeout in [ms] is the maximum wait time; after that if no switch has happened, the operation will fail.

#TPCMD GET_IVT <startAddress>

Used to get information on the IVT (proper source firmware file required). Start address is where to start finding the IVT.

#TPCMD GET_PART_SIZE <partitionName>

Used to get information on the size indicated in <partitionName> parameter.

Additional Parameters

#TCSETPAR SKIP_DCD

Used to skip dcd write during SDP boot/write command

#TCSETPAR SKIP_SPL

Used to skip secondary loader flashing during SDP/SDPU/SDPV write command