eMMC FLASH Programming User’s Guide

Release 09.2024

eMMC FLASH Programming User’s Guide

eMMC FLASH Programming User’s Guide

Version 05-Oct-2024

Introduction

This manual describes the basic concept of eMMC Flash programming.

How This Manual is Organized

- **About eMMC Interface Controllers in eMMC Flash Memories**: Provides background information about the topic.
- **Standard Approach**: Describes the fastest way to get started with eMMC Flash programming. All you need to do is to identify and run the correct script.

  Demo scripts for eMMC Flash programming are available in the folder:

  ~~/demo/*<architecture>*/flash/*.cmm

  e.g. at91sam3u-emmc.cmm, omap3530-emmc.cmm, …
- **New Scripts for eMMC Controllers**: Describes how you can create a script if there is no demo script for the eMMC controller you are using.

Related Documents

A complete description of all eMMC Flash programming commands can be found in chapter **“FLASHFILE”** in **“General Commands Reference Guide F”** (general_ref_f.pdf).

The Lauterbach home page provides an up-to-date list of

- Supported Flash devices under:
  https://www.lauterbach.com/ylist.html
- Supported eMMC Flash controllers under:
  https://www.lauterbach.com/ylistnand.html

Contacting Support

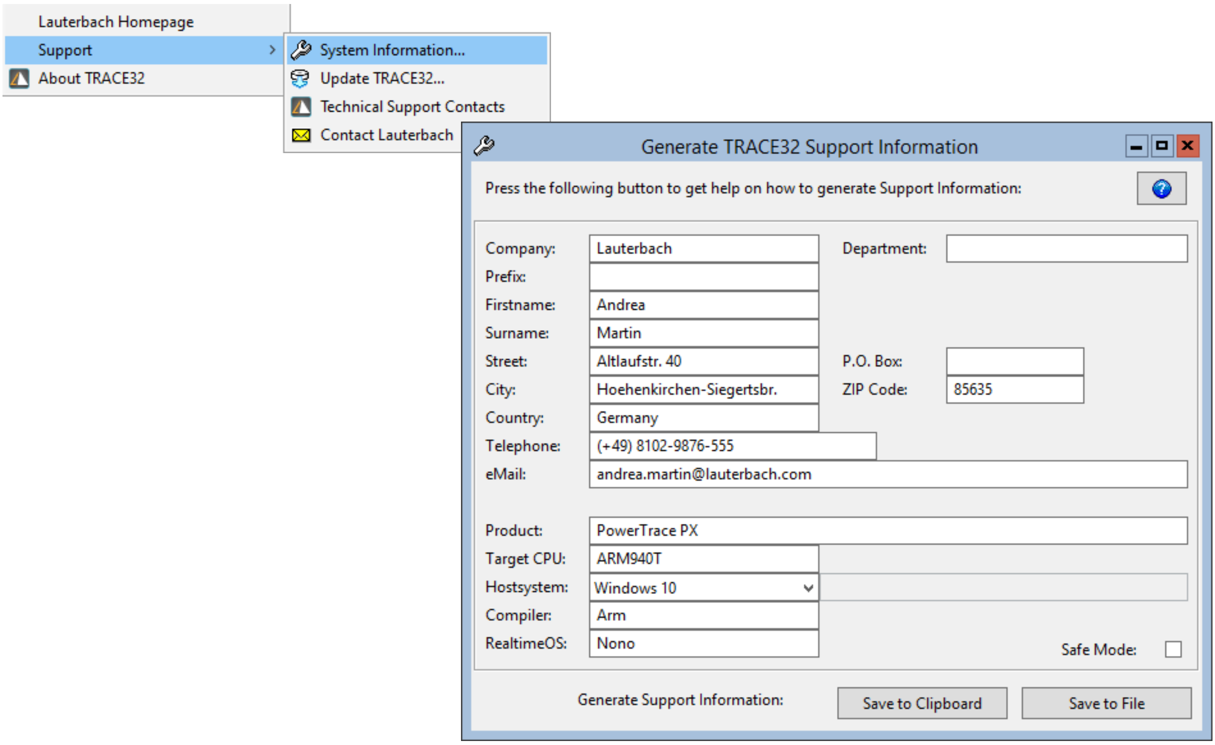
Use the Lauterbach Support Center: https://support.lauterbach.com

- To contact your local TRACE32 support team directly.
- To register and submit a support ticket to the TRACE32 global center.
- To log in and manage your support tickets.
- To benefit from the TRACE32 knowledgebase (FAQs, technical articles, tutorial videos) and our tips & tricks around debugging.

Or send an email in the traditional way to support@lauterbach.com.

Be sure to include detailed system information about your TRACE32 configuration.

1. To generate a system information report, choose **TRACE32** > **Help** > **Support** > **Systeminfo**.



Generate TRACE32 Support Information

Press the following button to get help on how to generate Support Information:

Company: Lauterbach
Department:
Prefix:
Firstname: Andrea
Surname: Martin
Street: Altlaufstr. 40
P.O. Box:
City: Hoehenkirchen-Siegertsbr.
ZIP Code: 85635
Country: Germany
Telephone: (+49) 8102-9876-555
eMail: andrea.martin@lauterbach.com

Product: PowerTrace PX
Target CPU: ARM940T
Hostsystem: Windows 10
Compiler: Arm
RealtimeOS: Nono
Safe Mode: □

Generate Support Information:

NOTE:	Please help to speed up processing of your support request. By filling out the system information form completely and with correct data, you minimize the number of additional questions and clarification request e-mails we need to resolve your problem.

2. Preferred: click **Save to File**, and send the system information as an attachment to your e-mail.

3. Click **Save to Clipboard**, and then paste the system information into your e-mail.

NOTE: In case of missing script files (`*.cmm`), please proceed as requested in “**If There is No Script**”.

List of Abbreviations

CS	Chip selection
eMMC	Embedded multimedia card
GPIO	General purpose input/output
JEDEC	Joint Electron Device Engineering Council
MISO	Master input, slave output
MMC	Multimedia card
MMCA	MultiMediaCard Association
MOSI	Master output, slave input
SCLK	Serial clock
SDI	Serial data input
SDO	Serial data output
SPI	Serial peripheral interface
SS	Slave select
SSI	Synchronous serial interface

Background Knowledge

This chapter of the manual is aimed at users who are new to eMMC Flash programming; it does not address experts with many years of expertise in this area. This chapter gives you a brief overview of important terms in eMMC Flash programming, such as eMMC interface controller, sector, and block.

What is an eMMC Flash Device?

An eMMC Flash device is a non-volatile, rewritable mass storage device. It is used in consumer products such as mobile phones, PDAs, and digital cameras. Reasons why eMMC Flash devices have become widespread include:

- High data storage capacity
- Easy to integrate into the host system
- The eMMC standard developed by the MMCA and the JEDEC is an open, royalty-free standard.

About Blocks and Pages

Sector A sector has a size of 512 bytes, the same size as a sector in the FAT file system under DOS.

Block A page is the minimum size unit for writing and reading. The size is configurable (512, 1024, 2048 bytes), but normally the size is 512 bytes.

About eMMC Interface Controllers in eMMC Flash Memories

eMMC Flash memories include an interface controller and a Flash memory. Access to the Flash memory is performed by the interface controller on the slave side.

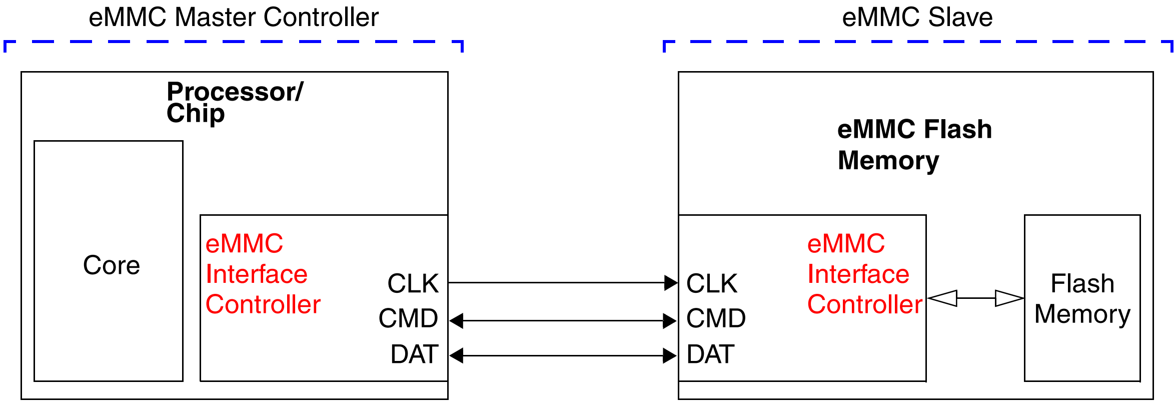


Figure: Processor/Chip and eMMC Flash Memory with an eMMC Interface

The protocol of the eMMC interface has three communication signals:

- MCC clock (CLK)
- Command in / response out (CMD)
- Data input / output (DAT)

Most chip manufacturers have their own MMC interface controllers (short: MMC controllers). As a result, MMC controllers require special driver binary files for programming eMMC Flash memories. These driver binary files are provided by Lauterbach.

Once the required driver binary file has been loaded to the target board, the eMMC Flash memory can be programmed and erased using the **FLASHFILE** command group in TRACE32.

Standard Approach

Standard Approach provides a compact description of the steps required to program eMMC Flash memories. This description is intentionally restricted to the standard use case.

For a detailed description of the eMMC Flash programming concepts, see “**Scripts for eMMC Controllers**” on **page 13**.

Identifying and Running Scripts for eMMC Flash Programming

Demo scripts (*.cmm) for eMMC Flash programming are provided by Lauterbach. They can be found in the TRACE32 installation directory. It contains scripts for programming eMMC Flash memories.

Path and file name convention of scripts to be used with eMMC controllers:

~~/demo/*<architecture>*/flash/*<cpu_name>*-*<emmc_flash_code>*.cmm
Where ~~ is expanded to the *<trace32_installation_directory>*, which is c:/t32 by default.

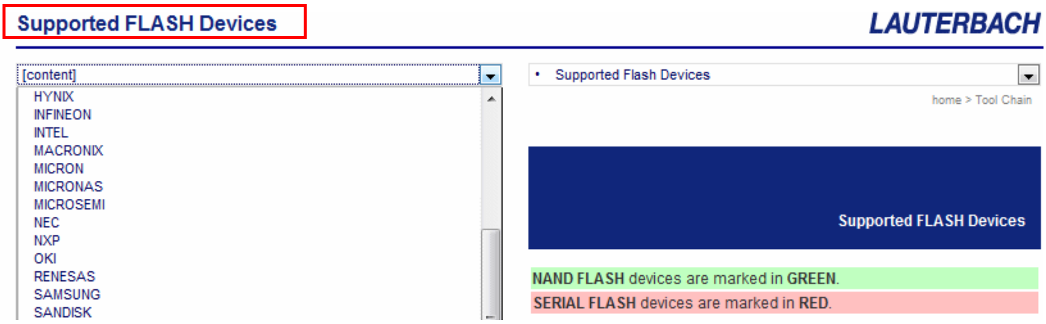
To identify and run the required script:

1. Make a note of the *<cpu_name>* printed on the CPU; for example, `dm365`.

2. For information about supported Flash devices, access the **Lauterbach** website.

3. Click the **+** tree button next to **Tool Chain**, and then click **Supported Flash Devices** (**https://www.lauterbach.com/ylist.html**).

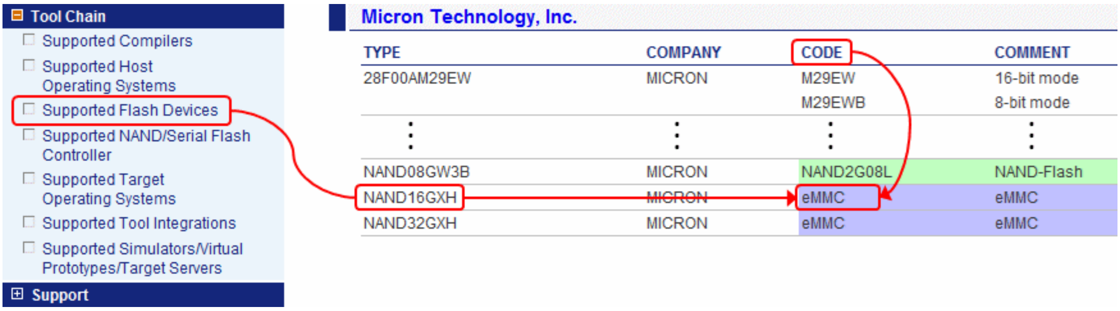
4. On the **Supported Flash Devices** page, select the required company from the drop-down list.



NAND FLASH devices are marked in GREEN.
SERIAL FLASH devices are marked in RED.

5. Use the type printed on the Flash device to retrieve the *<emmc_flash_code>* from the web page.

   For example, eMMC Flash type = NAND16GXH


Tool Chain
Supported Compilers
Supported Host Operating Systems
Supported Flash Devices
Supported NAND/Serial Flash Controller
Supported Target Operating Systems
Supported Tool Integrations
Supported Simulators/Virtual Prototypes/Target Servers
Support

Micron Technology, Inc.

TYPE	COMPANY	CODE	COMMENT
28F00AM29EW	MICRON	M29EW	16-bit mode
		M29EWB	8-bit mode
⋮	⋮	⋮	⋮
NAND08GW3B	MICRON	NAND2G08L	NAND-Flash
NAND16GXH	MICRON	eMMC	eMMC
NAND32GXH	MICRON	eMMC	eMMC

   Result: *<emmc_flash_code>* = emmc

6. Put the *<cpu_name>* and the *<emmc_flash_code>* together to form the script name: **`dm365-emmc`**`.cmm`

   The script resides in this folder: ~~/demo/arm/flash/dm365-emmc.cmm

   Where ~~ is expanded to the *<trace32_installation_directory>*, which is c:/t32 by default.

   If the folder does not contain the script you are looking for, see "**Scripts for eMMC Controllers**" on **page 13**.

7. Run the script in TRACE32 by doing one of the following:

   - Choose **File** > **Run Script** *<cmm_script_name>*

   - In the command line, type **DO** *<cmm_script_name>*

NOTE: Each script (*.cmm) includes a reference to the required eMMC Flash programming algorithm (*.bin).
You do not need to program or select the algorithm.

Example

```
;                      <code_range>        <data_range>        <file>
FLASHFILE.TARGET 0x20000000++0x1FFF 0x20002000++0x1FFF
                             ~~/demo/arm/flash/byte/emmc_at91sam.bin
```

If There Is No Script

If there is no script for your device in this directory (~~/demo/*<architecture>*/flash/), then please send a request to **support@lauterbach.com** using the e-mail template below.

E-Mail Template:

Chip name: ________________________

Name of serial Flash device: _________

Provide the CPU datasheet for us: ____________

Lend the target board to us by sending it to the address given in "**Contacting Support**": _________

<system_information>

Be sure to include detailed system information about your TRACE32 configuration. For information about how to create a system information report, see "**Contacting Support**".

Normally we can provide support for a new device in two weeks.

If our support cannot provide you with a PRACTICE script, you will have to create your own PRACTICE script (*.cmm).

For more information, see "**Scripts for eMMC Controllers**" on **page 13**.

Scripts for eMMC Controllers

This chapter describes how to create new scripts for eMMC Flash memories that are equipped with eMMC controllers.

The steps and the framework (see below) provide an overview of the process. They are described in detail in the following sections.

The following steps are necessary to create a new script:

1. Establish communication between debugger and target CPU.
2. Configure the eMMC controller.
3. Reset the eMMC Flash environment in TRACE32 to its default values.
4. Inform TRACE32 about the eMMC Flash register addresses (Flash declaration).
5. Inform TRACE32 about the eMMC Flash programming algorithm.
6. Check the identification from the eMMC Flash device.
7. Erase the eMMC Flash device.
8. Program the eMMC Flash device.

The following framework can be used as base for eMMC Flash programming:

```
                                        ; Establish the communication
                                        ; between the target CPU and the
                                        ; TRACE32 debugger.

                                        ; Configure the eMMC controller.

FLASHFILE.RESet                         ; Reset the eMMC Flash environment
                                        ; in TRACE32 to its default values.

FLASHFILE.CONFIG …                      ; Specify the base address of the
                                        ; control register of the eMMC
                                        ; controller.

FLASHFILE.TARGET …                      ; Inform Trace 32 about:
                                        ; - the FLASH programming algorithm
                                        ; - the <code_address> and
                                        ;   the <data_address> for the
                                        ;   FLASH programming algorithm

FLASHFILE.Erase …                       ; Erase the eMMC Flash.

FLASHFILE.LOAD <main_file> …            ; Program the file to eMMC Flash.
```

An ellipsis (…) in the framework indicates that command parameters have been omitted here for space economy.

NOTE: The parametrization of **FLASHFILE.CONFIG** and **FLASHFILE.TARGET** requires expert knowledge.

Establishing Communication between Debugger and Target CPU

eMMC Flash programming with TRACE32 requires that the communication between the debugger and the target CPU is established. The following commands are available to set up this communication:

SYStem.CPU *<cpu>*	Specify your target CPU.
SYStem.Up	Establish the communication between the debugger and the target CPU.

```
SYStem.CPU AT91SAM3U4         ; Select AT91SAM3U4 as the target CPU.

SYStem.Up                     ; Establish the communication between the
                              ; debugger and the target CPU.
```

Configuring the eMMC Controller

Programming a eMMC Flash device requires an appropriate initialization of the eMMC Flash interface. The following settings might be necessary:

- Enable the clock (CLK).
- Configure the registers of the eMMC Flash interface, such as clock, master/slave, data width, etc.
- Configure the eMMC Flash pins if they are muxed with other functions of the CPU.

Example

```
Data.Set 0x400E1254 %LE %Long 0x3FFFFFFF    ; Disable watchdog.
                                            ; LE = little endian

Data.Set 0x400E0C04 %LE %Long 0x1F8         ; Switch from the GPIO.A
                                            ; pins to the eMMC pins.

Data.Set 0x400E0430 %LE %Long 0x11          ; Enable the eMMC clock.
Data.Set 0x400E0410 %LE %Long 0x20000

Data.Set 0x40000008 %LE %Long 0x7F          ; Configure the eMMC
Data.Set 0x4000000C %LE %Long 0x00          ; controller (for example,
Data.Set 0x40000054 %LE %Long 0x1           ; bus width, clock speed and
Data.Set 0x40000004 %LE %Long 0x73B         ; time-out).

Data.Set 0x40000000 %LE %Long 0x1           ; Enable the eMMC
                                            ; controller.
```

Resetting Default Values

The following command is used to reset the eMMC Flash environment in TRACE32 to its default values.

FLASHFILE.RESet	Reset the eMMC Flash environment in TRACE32 to its default values.

Informing TRACE32 about the eMMC Controller Address

The following command is used to inform TRACE32 about the base address of the eMMC controller.

FLASHFILE.CONFIG *<mmc_controller_base_address> , ,*	, represents don't-care parameters.

For information about the base address, refer to the manufacturer's data sheet.

Example

```
; Base address of the eMMC controller in the AT91SAM3U
FLASHFILE.CONFIG 0x40000000 , ,
```

Informing TRACE32 about the eMMC Flash Programming Algorithm

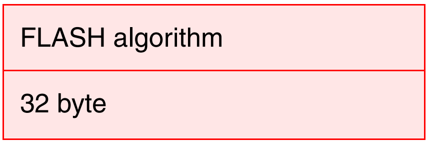
The following command is available to inform TRACE32 about the eMMC Flash programming algorithm:

FLASHFILE.TARGET *<code_range> <data_range> <file>*	Specify the eMMC Flash programming driver and where it runs in the target RAM.

Parameters

- *<code_range>*

  Define an address range in the target´s RAM to which the eMMC Flash programming algorithm is loaded.



FLASH algorithm

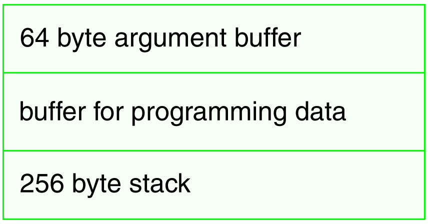
32 byte

Figure: Memory mapping for the *<code_range>*

Required size for the code is: size_of(*<file>*) + 32 byte

- *<data_range>*

  Define the address range in the target´s RAM where the programming data is buffered for the programming algorithm.

  64 byte argument buffer

  buffer for programming data

  256 byte stack

  Figure: Memory mapping for the *<data_range>*

  The argument buffer used for the communication between the TRACE32 software and the programming algorithm is located at the first 64 bytes of *<data_range>*. The 256 byte stack is located at the end of *<data_range>*.

  <buffer_size> =
  size_of(*<data_range>*) - 64 byte argument buffer - 256 byte stack

  <buffer_size> is the maximum number of bytes that are transferred from the TRACE32 software to the eMMC Flash programming algorithm in one call.

- *<file>*

  Lauterbach provides ready-to-run driver binary files for eMMC Flash programming. They are located in the TRACE32 installation directory:

  ~~/demo/*<architecture>*/flash/byte

  Where ~~ is expanded to the TRACE32 installation directory, which is c:/t32 by default.

  For detailed information about how to determine the *<file>* parameter, see “**Identifying the Correct Driver Binary File for an eMMC Flash Device**” on **page 18**.

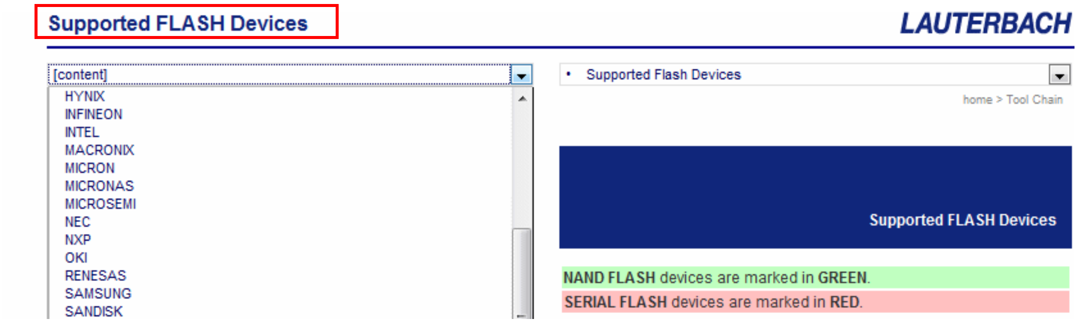
Identifying the Correct Driver Binary File for an eMMC Flash Device

1. For information about supported Flash devices, access the **Lauterbach** website.

2. Click the **+** tree button next to **Tool Chain**, and then click **Supported NAND/Serial Flash Controller** (**https://www.lauterbach.com/ylistnand.html**).

3. Open **Supported Flash Devices** in a separate window or tab (**https://www.lauterbach.com/ylist.html**).

4. On the **Supported Flash Devices** page, select the required company from the drop-down list.



NAND FLASH devices are marked in GREEN.

SERIAL FLASH devices are marked in RED.

5. Locate the desired Flash device.

   You need the name of the Flash device to be able to identify the correct driver binary file.

   The file name convention for driver binary files is explained below. In addition, an example illustrates how to apply the file name convention in practice.

File Name Convention for eMMC Flash Drivers

eMMC Flash drivers for eMMC controllers use the following file name convention:

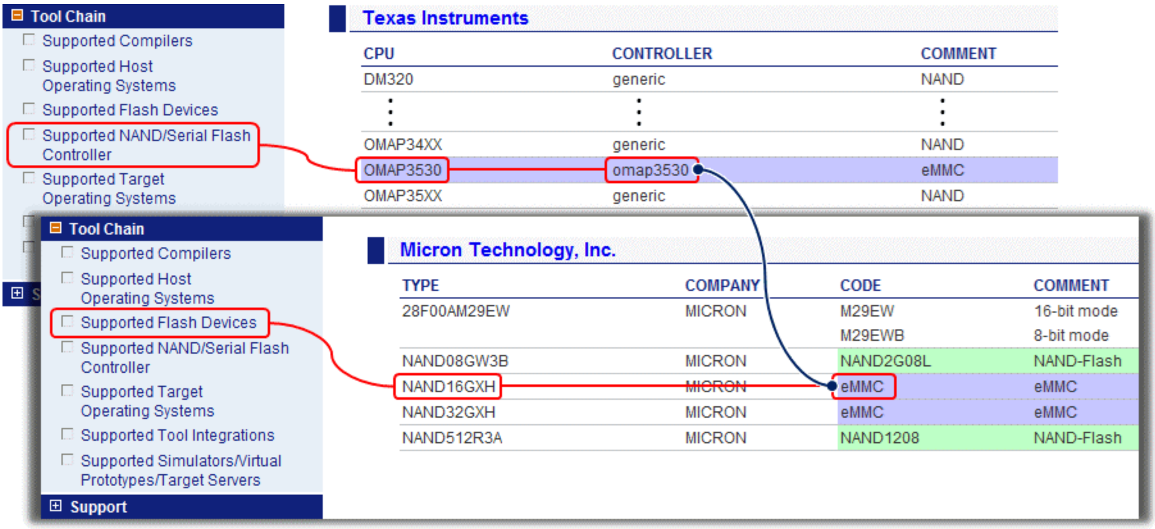
`eMMC_CPU.bin`
where **`CPU`** is the CPU family name.

Example for eMMC Controllers

Target:

- CPU **OMAP3530** with the eMMC controller **omap3530**
- eMMC Flash device **NAND16GXH**

Taken together, the **Code** column and the **Controller** column make up the file name of the eMMC Flash driver binary file: **eMMC_omap.bin**.


Tool Chain
Supported Compilers
Supported Host Operating Systems
Supported Flash Devices
Supported NAND/Serial Flash Controller
Supported Target Operating Systems
Supported Tool Integrations
Supported Simulators/Virtual Prototypes/Target Servers
Support

Texas Instruments

CPU	CONTROLLER	COMMENT
DM320	generic	NAND
⋮	⋮	⋮
OMAP34XX	generic	NAND
OMAP3530	omap3530	eMMC
OMAP35XX	generic	NAND

Micron Technology, Inc.

TYPE	COMPANY	CODE	COMMENT
28F00AM29EW	MICRON	M29EW	16-bit mode
		M29EWB	8-bit mode
NAND08GW3B	MICRON	NAND2G08L	NAND-Flash
NAND16GXH	MICRON	eMMC	eMMC
NAND32GXH	MICRON	eMMC	eMMC
NAND512R3A	MICRON	NAND1208	NAND-Flash

The binary file resides in this folder: ~~/demo/arm/flash/byte

Where ~~ is expanded to the TRACE32 installation directory, which is c:/t32 by default.

This results in the following command line:

```
; Specify the eMMC Flash programming algorithm and where it runs in
; the target RAM.    <code_range>        <data_range>          <file>
FLASHFILE.TARGET 0x4020000++0x1FFF 0x4022000++0x1FFF
                                  ~~/demo/arm/flash/byte/emmc_omap.bin
```

FLASHFILE Declaration Examples

Declaration Example 1

CPU:	OMAP4430 (Texas Instruments)
Base address of the eMMC controller:	0x480B4000
Driver file:	~~/demo/arm/flash/byte/emmc_omap.bin Where ~~ is expanded to the TRACE32 installation directory, which is c:/t32 by default.

```
…

; Reset the FLASHFILE declaration within TRACE32.
FLASHFILE.RESet

; Base address of the eMMC controller in the OMAP4430
FLASHFILE.CONFIG 0x480B4000 , ,

; Specify the eMMC Flash programming algorithm and where it runs on
; the target RAM.  <code_range>       <data_range>        <file>
FLASHFILE.TARGET 0x40301000++0x1FFF 0x40303000++0x2FFF
                               ~~/demo/arm/flash/byte/emmc_omap.bin

…
```

Declaration Example 2

CPU:	AT91SAM3U4 (ATMEL)
Base address of the eMMC controller:	0x40000000
Driver file:	`~~/demo/arm/flash/byte/emmc_at91sam.bin` Where ~~ is expanded to the TRACE32 installation directory, which is c:/t32 by default.

```
…

; Reset the FLASHFILE declaration within TRACE32.
FLASHFILE.RESet

; Base address of the eMMC controller in the AT91SAM3U4
FLASHFILE.CONFIG 0x40000000 , ,

; Specify the eMMC Flash programming algorithm and where it runs on
; the target RAM.  <code_range>      <data_range>          <file>
FLASHFILE.TARGET 0x20000000++0x1FFF 0x20002000++0x1FFF
                                 ~~/demo/arm/flash/byte/emmc_at91sam.bin

…
```

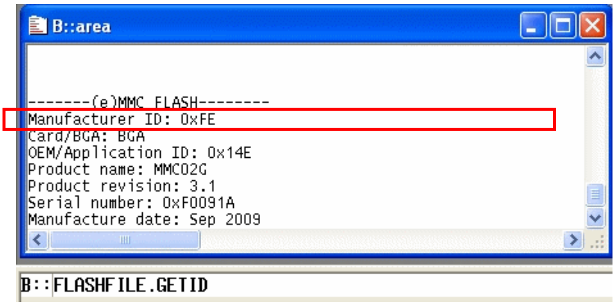
Checking the Identification from the eMMC Flash Device

The following command can be used to check if TRACE32 can access the eMMC Flash device:

FLASHFILE.GETID	Get the ID values of the eMMC Flash device.

```
; Open the TRACE32 AREA window.
AREA.view

; Check the access to the eMMC Flash device
; by getting the manufacturer ID and the device ID.
FLASHFILE.GETID
```



```
-------(e)MMC FLASH--------
Manufacturer ID: 0xFE
Card/BGA: BGA
OEM/Application ID: 0x14E
Product name: MMC02G
Product revision: 3.1
Serial number: 0xF0091A
Manufacture date: Sep 2009
```

```
B::FLASHFILE.GETID
```

Erasing the eMMC Flash Device

The following command is available to erase eMMC Flash devices:

FLASHFILE.Erase *<range>*	Erase a specified range of the eMMC Flash device.

Example:

```
; Erase 2MB starting at 0x0.
FLASHFILE.Erase 0x0--0x1FFFFF
```

Programming the eMMC Flash Device

The following commands are available to program the eMMC Flash:

FLASHFILE.LOAD *<file>* [*<address>* | *<range>*]	Program the eMMC Flash.
FLASHFILE.LOAD *<file>* [*<address>* | *<range>*] **/ComPare**	Verify the contents of the file against the eMMC Flash.

The data from *<file>* is written to the address range specified by *<range>*. If no *<range>* or *<address>* is specified, programming starts at address 0x0.

Example 1:

```
; Program the contents of my_file.bin to the eMMC Flash memory starting
; at address 0x0.
FLASHFILE.LOAD my_file.bin 0x0
```

Example 2:

```
; Verify the contents of my_file.bin against the eMMC Flash memory
; starting at address 0x0.
FLASHFILE.LOAD my_file.bin 0x0 /ComPare
```

Copying the eMMC Flash Memory

The following command is available to copy:

- Any data from any CPU memory area to the eMMC Flash memory, or
- Any data from one address range of the eMMC Flash to another address range within the same eMMC Flash memory; for example, for backup purposes.

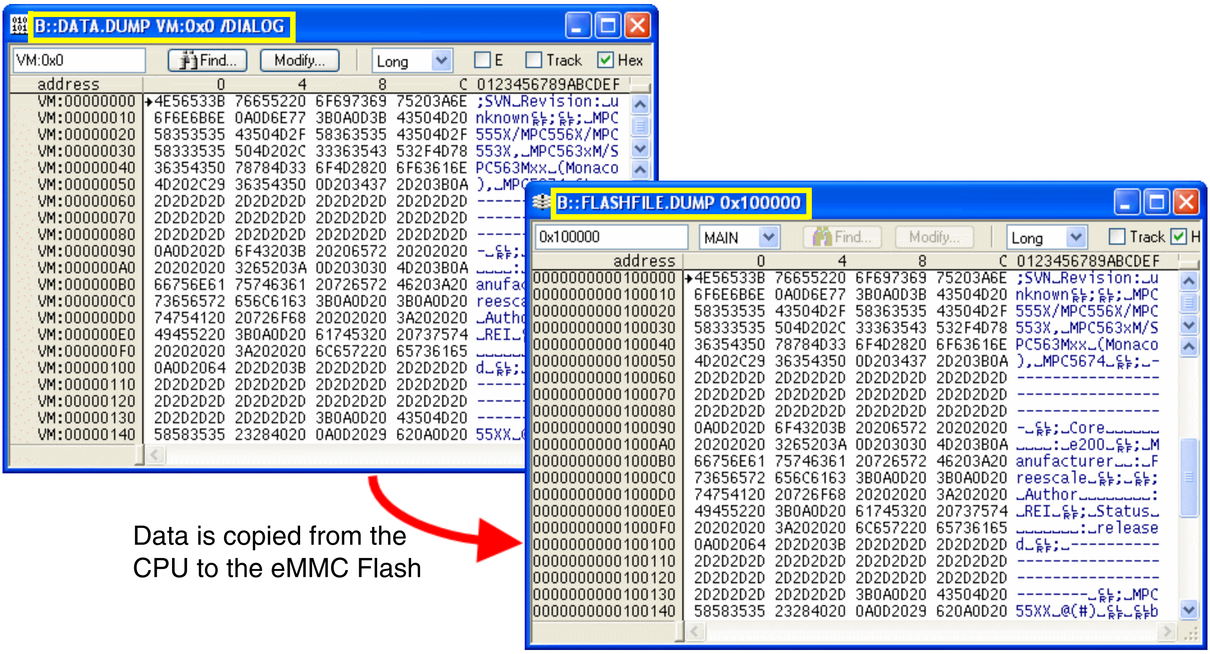
FLASHFILE.COPY <*source range*> <*target addr*>	Copy data from the source range to the defined address of the eMMC Flash.

FLASHFILE.COPY <*source range*> <*target addr*> **/ComPare**	Verify the source range data against the target range data.

Example 1:

```
; Copy the 1MB virtual memory data at 0x0 to the eMMC Flash address
; at 0x100000.
; VM: The virtual memory of the TRACE32 software.
FLASHFILE.COPY VM:0x0--0xFFFFF 0x100000
```

Result (1):



B::DATA.DUMP VM:0x0 /DIALOG

```
    address          0        4        8        C 0123456789ABCDEF
VM:00000000 ▸4E56533B 76655220 6F697369 75203A6E ;SVN␣Revision:␣u
VM:00000010  6F6E6B6E 0A0D6E77 3B0A0D3B 43504D20 nknown␍␊;␍␊;␣MPC
VM:00000020  58353535 43504D2F 58363535 43504D2F 555X/MPC556X/MPC
VM:00000030  58333535 504D202C 33363543 532F4D78 553X,␣MPC563xM/S
VM:00000040  36354350 78784D33 6F4D2820 6F63616E PC563Mxx␣(Monaco
VM:00000050  4D202C29 36354350 0D203437 2D203B0A ),␣MPC
VM:00000060  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ------
VM:00000070  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ------
VM:00000080  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ------
VM:00000090  0A0D202D 6F43203B 20206572 20202020 -␣␍␊;
VM:000000A0  20202020 3265203A 0D203030 4D203B0A ␣␣␣␣:
VM:000000B0  66756E61 75746361 20726572 46203A20 anufac
VM:000000C0  73656572 656C6163 3B0A0D20 3B0A0D20 reesca
VM:000000D0  74754120 20726F68 20202020 3A202020 ␣Autho
VM:000000E0  49455220 3B0A0D20 61745320 20737574 ␣REI␣
VM:000000F0  20202020 3A202020 6C657220 65736165 ␣␣␣␣␣␣
VM:00000100  0A0D2064 2D2D203B 2D2D2D2D 2D2D2D2D d␣␍␊;
VM:00000110  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ------
VM:00000120  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ------
VM:00000130  2D2D2D2D 2D2D2D2D 3B0A0D20 43504D20 ------
VM:00000140  58583535 23284020 0A0D2029 620A0D20 55XX␣
```

B::FLASHFILE.DUMP 0x100000

```
         address          0        4        8        C 0123456789ABCDEF
0000000000100000 ▸4E56533B 76655220 6F697369 75203A6E ;SVN␣Revision:␣u
0000000000100010  6F6E6B6E 0A0D6E77 3B0A0D3B 43504D20 nknown␍␊;␍␊;␣MPC
0000000000100020  58353535 43504D2F 58363535 43504D2F 555X/MPC556X/MPC
0000000000100030  58333535 504D202C 33363543 532F4D78 553X,␣MPC563xM/S
0000000000100040  36354350 78784D33 6F4D2820 6F63616E PC563Mxx␣(Monaco
0000000000100050  4D202C29 36354350 0D203437 2D203B0A ),␣MPC5674␣␍␊;␣-
0000000000100060  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ----------------
0000000000100070  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ----------------
0000000000100080  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ----------------
0000000000100090  0A0D202D 6F43203B 20206572 20202020 -␣␍␊;␣Core␣␣␣␣␣␣
00000000001000A0  20202020 3265203A 0D203030 4D203B0A ␣␣␣␣:␣e200␣␍␊;␣M
00000000001000B0  66756E61 75746361 20726572 46203A20 anufacturer␣␣:␣F
00000000001000C0  73656572 656C6163 3B0A0D20 3B0A0D20 reescale␣␍␊;␣␍␊;
00000000001000D0  74754120 20726F68 20202020 3A202020 ␣Author␣␣␣␣␣␣␣␣:
00000000001000E0  49455220 3B0A0D20 61745320 20737574 ␣REI␣␍␊;␣Status␣
00000000001000F0  20202020 3A202020 6C657220 65736165 ␣␣␣␣␣␣␣:␣release
0000000000100100  0A0D2064 2D2D203B 2D2D2D2D 2D2D2D2D d␣␍␊;␣----------
0000000000100110  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ----------------
0000000000100120  2D2D2D2D 2D2D2D2D 2D2D2D2D 2D2D2D2D ----------------
0000000000100130  2D2D2D2D 2D2D2D2D 3B0A0D20 43504D20 --------␣␍␊;␣MPC
0000000000100140  58583535 23284020 0A0D2029 620A0D20 55XX␣@(#)␣␍␊␣␍␊b
```

Data is copied from the CPU to the eMMC Flash

Example 2:

```
; Verify the data between virtual memory and eMMC Flash.
FLASHFILE.COPY VM:0x0--0xFFFFF 0x100000 /ComPare
```

Example 3:

```
; Copy the 1MB eMMC Flash data at 0x0 to the eMMC Flash
; at 0x800000.
FLASHFILE.COPY 0x0--0xFFFFF 0x800000

; Verify the 1MB eMMC Flash data between 0x0 and 0x800000.
FLASHFILE.COPY 0x0--0xFFFFF 0x800000 /ComPare
```

Modifying the eMMC Flash Memory

The following command is available to modify the contents of the eMMC Flash memory.

FLASHFILE.Set [*<address>* | *<range>*] **%***<format>* *<data>*	Modify the contents of the eMMC Flash.

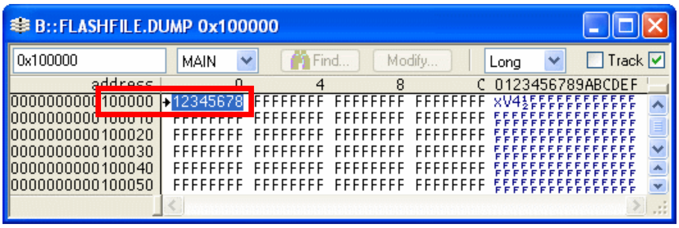
Example 1:

```
; Write 4 bytes of data 0x12345678 to the address 0x100000.
; LE = little endian
FLASHFILE.Set 0x100000 %LE %Long 0x12345678
```

Example 2:

```
; Write data 0x0 from 0x100000 to 0x13FFFF in the eMMC Flash.
FLASHFILE.Set 0x100000++0x13FFFF %Long 0x0
```

Result (1)



B::FLASHFILE.DUMP 0x100000

0x100000 | MAIN | Find... | Modify... | Long | ☐ Track ☑

```
         address        0        4        8        C  0123456789ABCDEF
0000000000100000 12345678 FFFFFFFF FFFFFFFF FFFFFFFF
0000000000100010 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
0000000000100020 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
0000000000100030 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
0000000000100040 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
0000000000100050 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
```

Other Useful Commands

The CPU cannot read eMMC Flash memories directly. But TRACE32 provides special commands for reading eMMC Flash memories. The contents of the eMMC Flash are displayed in a window.

Reading the eMMC Flash

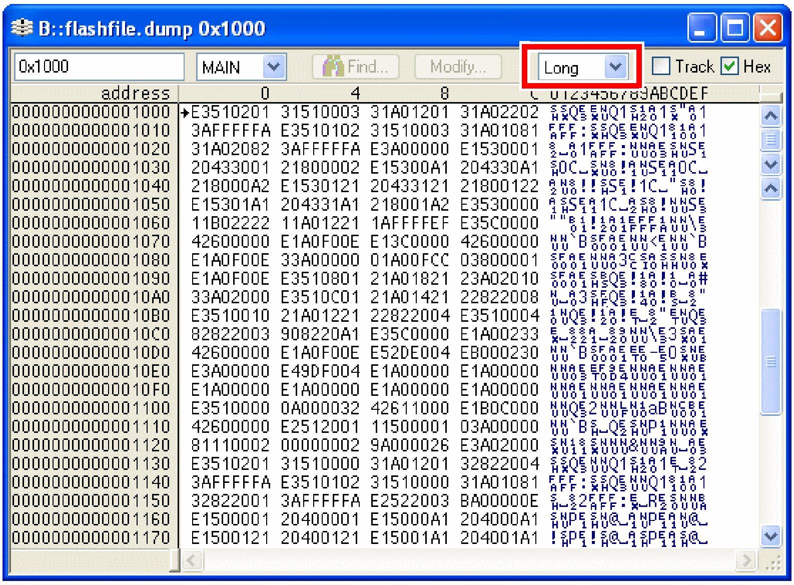
The following command allows to read the eMMC Flash memory.

FLASHFILE.DUMP [*<address>*] [**/***<format>*]	Display a hex-dump of the eMMC Flash.

Example:

```
; Display a hex-dump of the eMMC Flash starting at 0x1000.
; Display the information in 2-bit format (/Long option).
FLASHFILE.DUMP 0x1000 /Long
```

Result



B::flashfile.dump 0x1000

```
         address         0        4        8        C  0123456789ABCDEF
0000000000001000 ►E3510201 31510003 31A01201 31A02202
0000000000001010  3AFFFFFA E3510102 31510003 31A01081
0000000000001020  31A02082 3AFFFFFA E3A00000 E1530001
0000000000001030  20433001 21800002 E15300A1 204330A1
0000000000001040  218000A2 E1530121 20433121 21800122
0000000000001050  E15301A1 204331A1 218001A2 E3530000
0000000000001060  11B02222 11A01221 1AFFFFEF E35C0000
0000000000001070  42600000 E1A0F00E E13C0000 42600000
0000000000001080  E1A0F00E 33A00000 01A00FCC 03800001
0000000000001090  E1A0F00E E3510801 21A01821 23A02010
00000000000010A0  33A02000 E3510C01 21A01421 22822008
00000000000010B0  E3510010 21A01221 22822004 E3510004
00000000000010C0  82822003 908220A1 E35C0000 E1A00233
00000000000010D0  42600000 E1A0F00E E52DE004 EB000230
00000000000010E0  E3A00000 E49DF004 E1A00000 E1A00000
00000000000010F0  E1A00000 E1A00000 E1A00000 E1A00000
0000000000001100  E3510000 0A000032 42611000 E1B0C000
0000000000001110  42600000 E2512001 11500001 03A00000
0000000000001120  81110002 00000002 9A000026 E3A02000
0000000000001130  E3510201 31510000 31A01201 32822004
0000000000001140  3AFFFFFA E3510102 31510000 31A01081
0000000000001150  32822001 3AFFFFFA E2522003 BA00000E
0000000000001160  E1500001 20400001 E15000A1 204000A1
0000000000001170  E1500121 20400121 E15001A1 204001A1
```

Saving the eMMC Flash Device

The following command is available to save the contents of the eMMC Flash memory to a file.

FLASHFILE.SAVE *<file> <range>*	Save the contents of the eMMC Flash memory into *<file>.*

Example:

```
; Save 1MB of the eMMC Flash data starting at 0x0 to the file
; my_dump.bin.
FLASHFILE.SAVE my_dump.bin 0x0--0xFFFFF
```

Full Examples

Example 1

CPU:	OMAP4430
eMMC Flash:	SanDisk, iNAND 8 GBytes
Internal SRAM:	0x40301000
SD/MMC Controller Register:	0x480B4000 (MMCHS2)

```
SYStem.RESet

system.CPU omap4430

SYStem.JtagClock 10.Mhz

SYStem.Option.DACR ON  ; Give debugger global write permissions.

SETUP.IMASKASM OFF     ; Lock interrupts while single stepping.

SYSTEM.MEMACCESS DAP   ; Enable DAP access.

TrOnchip.set DABORT OFF

TrOnchip.set PABORT OFF

TrOnchip.Set UNDEF OFF

SYStem.mode.attach

wait 1.s

if run()

break

  GOSUB disable_watchdog

  D.S NSD:0x480B422C %LE %Long 0xe0f87   ; Configure the eMMC clock.
                                         ; LE = little endian

  BREAK.RESet

  FLASHFILE.RESet


  ; Base address of the eMMC controller in the OMAP4430
  FLASHFILE.CONFIG 0x480B4000 , ,
; Specify the eMMC Flash programming algorithm and where it runs in
; the target RAM.   <code_range>      <data_range>         <file>
  FLASHFILE.TARGET 0x40301000++0x1FFF 0x40303000++0x1FFF
                                ~~/demo/arm/flash/byte/emmc_omap.bin

; Check the access to the eMMC Flash device
; by getting the manufacturer ID and the device ID.
  FLASHFILE.GETID

  DIALOG.YESNO “Program flash memory?”

  ENTRY &progflash

  IF &progflash

  (; Erase Flash.

    FLASHFILE.ERASE 0x0--0xFFFFFF

  ; Write Flash.
    FLASHFILE.LOAD * 0x0

  ; Verify Flash.
    FLASHFILE.LOAD * 0x0 /ComPare

  )

  ; Display a hex-dump of the eMMC Flash starting at 0x0.
  FLASHFILE.DUMP 0x0

ENDDO
```

Example 2

CPU:	DM365
eMMC Flash:	Numonyx, NAND16GXH is connected to the MMC0 controller.
SDRAM:	0x80002000
eMMC Controller Register:	0x01D11000

```
SYStem.Down
SYStem.JtagClock 1Mhz

SYStem.RESet
SYStem.CPU DM365
SYStem.o.rb off
SYStem.JtagClock 1Mhz
SYStem.mode go
wait 1.s
if run()
break

; Enable for the eMMC.
Data.Set 0x1C48018 %Long 0x4000000  ; Enable for MMC0 controller

; Configure eMMC CLK.
Data.Set 0x01D11004  %Long 0x0117  ; MMC CLK
Data.Set 0x01D11000  %Long 0x0007  ; MMC_CTL
Data.Set 0x01D11000  %Long 0x0000  ; MMC_CTL

  FLASHFILE.RESet
  Break.RESet

; Base address of the eMMC controller in the DM365.
  FLASHFILE.CONFIG 0x01D11000 , ,

; Specify the eMMC Flash programming algorithm and where it runs in
; the target RAM.   <code_range>      <data_range>        <file>
  FLASHFILE.TARGET 0x80002000++0x1FFF 0x80004000++0x1FFF
                                   ~~/demo/arm/flash/byte/emmc_dm365.bin

; Check the access to the eMMC Flash device
; by getting the manufacturer ID and the device ID.
  FLASHFILE.GETID

  DIALOG.YESNO "Program flash memory?"
  ENTRY &progflash
  IF &progflash

  (; Erase FLASH.

   FLASHFILE.ERASE 0x0--0xFFFFFF
  ; Write Flash.
    FLASHFILE.LOAD  * 0x0

  ; Verify Flash.
  FLASHFILE.LOAD  * 0x0  /ComPare

  )

  ; Display a hex-dump of the eMMC Flash starting at 0x0.
  FLASHFILE.DUMP 0x0

ENDDO
```

FLASH Programming via Boundary Scan

The BSDL commands of TRACE32 are used to program external FLASH memories via boundary scan. Important BSDL-specific steps are:

- Check that the bypass mode works.
- Check that the IDCODE matches.
- Define the FLASH pin connection.
- Enable eMMC FLASH programming via boundary scan and define the flash type.

eMMC FLASH programming then continues with the **FLASHFILE** commands described in this manual. The following PRACTICE script (*.cmm) illustrates the BSDL-specific steps by way of this example for the MMC protocol:

CPU:	AT91SAM3U4
eMMC FLASH:	Numonyx, NAND16GXH
Pin connection:	MMC_CLK: Port A3 MMC_CMD: Port A4 MMC_DAT0: Port A5

```
SYStem.JtagClock 15.Mhz           ; set JTAG clock
BSDL.RESet                        ; reset boundary scan configuration
BSDL.FILE ./sam3u4e_lqfp144.bsd   ; load the required BSDL file

BSDL.HARDRESET                    ; toggle TRST_N pin
BSDL.SOFTRESET                    ; do a sequential JTAG reset

IF BSDL.CHECK.BYPASS()            ; check, if BYPASS mode works
(
   IF BSDL.CHECK.IDCODE()         ; check, if the IDCODE matches
   (
    BSDL.FLASH.IFDefine RESet     ; reset the boundary scan flash
                                  ; configuration

    BSDL.FLASH.IFDefine MMC 1. 1. ; define boundary scan flash
                                  ; interface:
                                  ; - protocol: MMC
                                  ; - MMC flash memory connected to
                                  ;   IC1 of the boundary scan chain
                                  ; - data with is 1 bit

    BSDL.FLASH.IFMAP CLK  PA3     ; map generic MMC pin CLK to port PA3
    BSDL.FLASH.IFMAP CMD  PA4     ; map generic MMC pin CMD to port PA4
    BSDL.FLASH.IFMAP DAT0 PA5     ; map generic MMC pin DAT0 to port PA5
    BSDL.FLASH.INIT SAFE          ; Initialize boundary scan chain to
                                  ; safe values according to
                                  ; SAFE state from BSDL file

    FLASHFILE.BSDLaccess ON       ; Enable flash programming
                                  ; via boundary scan
    FLASHFILE.BSDLFLASHTYPE EMMC  ; define flash type

    FLASHFILE.GETID               ; get the MMC flash memory ID
    ; continue with flash programming, e.g.
    ; FLASHFILE.DUMP 0x0
    ; FLASHFILE.ERASE 0x0--0xFFFFF
    ; FLASHFILE.LOAD  * 0x0
   )
)
ENDDO
```