

# eMMC FLASH Programming User's Guide

Release 02.2025



TRACE32 Online Help	
TRACE32 Directory	
TRACE32 Index	
TRACE32 Documents	
FLASH Programming	
eMMC FLASH Programming User's Guide	1
Introduction	4
How This Manual is Organized	4
Related Documents	4
Contacting Support	5
List of Abbreviations	7
Background Knowledge	8
What is an eMMC Flash Device?	8
About Blocks and Pages	8
About eMMC Interface Controllers in eMMC Flash Memories	9
Standard Approach	10
Identifying and Running Scripts for eMMC Flash Programming	10
If There Is No Script	12
Scripts for eMMC Controllers	13
Establishing Communication between Debugger and Target CPU	14
Configuring the eMMC Controller	15
Resetting Default Values	16
Informing TRACE32 about the eMMC Controller Address	16
Informing TRACE32 about the eMMC Flash Programming Algorithm	16
Identifying the Correct Driver Binary File for an eMMC Flash Device	18
File Name Convention for eMMC Flash Drivers	18
Example for eMMC Controllers	19
FLASHFILE Declaration Examples	20
Declaration Example 1	20
Declaration Example 2	21
Checking the Identification from the eMMC Flash Device	22
Erasing the eMMC Flash Device	23
Programming the eMMC Flash Device	23
Copying the eMMC Flash Memory	24
Modifying the eMMC Flash Memory	26
Other Useful Commands	27

FLASH Programming via Boundary Scan	32
Example 2	31
Example 1	29
Full Examples	29
Saving the eMMC Flash Device	28
Reading the eMMC Flash	27

Version 13-Feb-2025

## 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: www.lauterbach.com/supported-platforms/toolchain/flash-devices
- Supported eMMC Flash controllers under: www.lauterbach.com/supported-platforms/toolchain/flash-controllers

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.

Support       > </th <th>Lauterbach Homepage</th> <th></th> <th></th> <th></th> <th></th> <th></th>	Lauterbach Homepage					
▲ About TRACE32       Update TRACE32         Image: Technical Support Contacts         Image: Contact Lauterback         Ima	Support >	Ø System Information				
Image: Contact Lauterback       Generate TRACE32 Support Information         Image: Contact Lauterback       Press the following button to get help on how to generate Support Information:         Image: Company:       Lauterback       Department:         Image: Prefix:       Firstname:       Andrea         Surname:       Martin       P.O. Box:       Biological         Street:       Altlaufstr. 40       P.O. Box:       Biological         Country:       Germany       Clauterback.com       Biological         Telephone:       (+49) 8102-9876-555       Biological       Biological         Product:       PowerTrace PX       Target CPU:       ARM940T         Hostsystem:       Windows 10       V       Safe Mode:         Compile:       Arm       RealtimeOS:       Nono       Safe Mode:	About TRACE32	😌 Update TRACE32				
☑ Contact Lauterback       ✓       Generate TRACE32 Support Information         Press the following button to get help on how to generate Support Information:       ✓         Company:       Lauterbach       Department:         Prefix:       —       —         Firstname:       Andrea       —         Sumame:       Martin       P.O. Box:         Street:       Altlaufstr. 40       P.O. Box:         Country:       Germany       ZIP Code:         Telephone:       (+49) 8102-9876-555		🔼 Technical Support Cor	ntacts			
Press the following button to get help on how to generate Support Information:       Image: Company:       Lauterbach       Department:       Image: Company:       Prefix:		🔀 Contact Lauterbach	Þ	Generate TRACE32 Su	upport Information	<b>– – ×</b>
Company:       Lauterbach       Department:         Prefix:			Press the fol	lowing button to get help on how to g	generate Support Information:	<b>@</b>
Firstname:       Andrea         Surname:       Martin         Street:       Altlaufstr. 40       P.O. Box:         City:       Hoehenkirchen-Siegertsbr.       ZIP Code:       85635         Country:       Germany       Image: Composition of the strength of the strengt of the strength of the strength of the str			Company:	Lauterbach	Department:	
Surname:       Martin         Street:       Altlaufstr. 40       P.O. Box:         City:       Hoehenkirchen-Siegertsbr.       ZIP Code:       85635         Country:       Germany       Telephone:       (+49) 8102-9876-555			Firstname:	Andrea		
Street:       Altlaufstr. 40       P.O. Box:         City:       Hoehenkirchen-Siegertsbr.       ZIP Code:       85635         Country:       Germany       1       1         Telephone:       (+49) 8102-9876-555       1       1         eMail:       andrea.martin@lauterbach.com       1       1         Product:       PowerTrace PX       1       1         Target CPU:       ARM940T       1       1         Hostsystem:       Windows 10       v       1         Compiler:       Arm       Safe Mode:       1         RealtimeOS:       Nono       Safe Mode:       1			Surname:	Martin		
City:       Hoehenkirchen-Siegertsbr.       ZIP Code:       85635         Country:       Germany       [+49) 8102-9876-555			Street:	Altlaufstr. 40	P.O. Box:	
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			City:	Hoehenkirchen-Siegertsbr.	ZIP Code: 85635	
Telephone:       (+49) 8102-9876-555         eMail:       andrea.martin@lauterbach.com         Product:       PowerTrace PX         Target CPU:       ARM940T         Hostsystem:       Windows 10 v         Compiler:       Arm         RealtimeOS:       Nono         Safe Mode:       Component Information			Country:	Germany	L	
eMail: andrea.martin@lauterbach.com  Product: PowerTrace PX  Target CPU: ARM940T  Hostsystem: Windows 10 v  Compiler: Arm  RealtimeOS: Nono Safe Mode: [			Telephone:	(+49) 8102-9876-555		
Product:       PowerTrace PX         Target CPU:       ARM940T         Hostsystem:       Windows 10       v         Compiler:       Arm         RealtimeOS:       Nono       Safe Mode:			eMail:	andrea.martin@lauterbach.com		
Product:       PowerTrace PX         Target CPU:       ARM940T         Hostsystem:       Windows 10 v         Compiler:       Arm         RealtimeOS:       Nono						
Target CPU:     ARM940T       Hostsystem:     Windows 10 v       Compiler:     Arm       RealtimeOS:     Nono   Safe Mode:			Product:	PowerTrace PX		
Hostsystem: Windows 10 v Compiler: Arm RealtimeOS: Nono Safe Mode:			Target CPU:	ARM940T		
Compiler: Arm RealtimeOS: Nono Safe Mode: [			Hostsystem:	Windows 10 🗸 🗸		
RealtimeOS: Nono Safe Mode: [			Compiler:	Arm		
Generate Support Information			RealtimeOS:	Nono		Safe Mode:
Save to File				Generate Support Information:	Save to Clipboard	Save to File

**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".

CS	Chip selection
eMMC	Embedded multimedia card
GPIO	General purpose input/output
JEDEC	Joint Electron Device Engineering Council
MISO	Master input, slave output
ММС	Multimedia card
ММСА	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

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.

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** (www.lauterbach.com/supported-platforms/toolchain/flash-devices).
- 4. On the Supported Flash Devices page, select the required company from the drop-down list.



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	Micron Technology, In	с.		
Supported Compilers     Supported Uset	ТҮРЕ	COMPANY	CODE	COMMENT
Operating Systems	28F00AM29EW	MICRON	M29EW	16-bit mode
Supported Flash Devices			M29EWB	8-bit mode
Supported NAND/Serial Flash Controller	:	:		:
Supported Target	NAND08GW3B	MICRON	NAND2G08L	NAND-Flash
Operating Systems	NAND16GXH	MICRON	eMMC	eMMC
Supported Tool Integrations	NAND32GXH	MICRON	eMMC	eMMC
<ul> <li>Supported Simulators/Virtual Prototypes/Target Servers</li> </ul>				
Support				

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 0x2000000++0x1FFF 0x20002000++0x1FFF
~~/demo/arm/flash/byte/emmc\_at91sam.bin

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.

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</data_address></code_address>
FLASHFILE.Erase	; Erase the eMMC Flash.
FLASHFILE.LOAD <main_file></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></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.

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.
```

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 \ensuremath{\texttt{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></file></data_range></code_range>	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.



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>*.

<br/><br/>size> =<br/>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 (www.lauterbach.com/supported-platforms/toolchain/flash-controllers)**.
- Open Supported Flash Devices in a separate window or tab (www.lauterbach.com/supported-platforms/toolchain/flash-devices).
- 4. On the **Supported Flash Devices** page, select the required company from the drop-down list.

Supported FLASH Devices	LAUTERB	ACH
[content] HYNX NFINEON INTEL	Supported Flash Devices	vol Chain
MACRONIX MICRONAS MICRONAS MICROSEMI NEC NXP OKI	Supported FLASH De	vices
RENESAS SAMSUNG SANDISK	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.

## 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	Texas Instruments			
Supported Compilers	CPU	CONTROLLER	COMMEN	т
Operating Systems	DM320	generic	NAND	
Supported Flash Devices		:		
Supported NAND/Serial Flash	OMAP34XX	generic	NAND	
Controller	OMAP3530	omap3530	eMMC	
Operating Systems	OMAP35XX	generic	NAND	
Tool Chain     Supported Compilers	Micron Technolo	gy, Inc.		
Supported Host	ТҮРЕ	COMPANY	CODE	COMMENT
Supported Flash Devices	28F00AM29EW	MICRON	M29EW M29EWB	16-bit mode 8-bit mode
Controller	NAND08GW3B	MICRON	NAND2G08L	NAND-Flash
Supported Target	NAND16GXH	MICRON		eMMC
Operating Systems	NAND32GXH	MICRON	eMMC	eMMC
Supported Tool Integrations	NAND512R3A	MICRON	NAND1208	NAND-Flash
Supported Simulators/Virtual Prototypes/Target Servers				

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
```

## **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

••••

•••

CPU:	AT91SAM3U4 (ATMEL)
Base address of the eMMC controller:	0x4000000
Driver file:	~~/demo/arm/flash/byte/emmc_at91sam.bin Where ~~ is expanded to the TRACE32 installation directory, which is c:/t32 by default.

## 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

; by getting the manufacturer ID and the device ID.

FLASHFILE.GETID

🖹 B::area 📃 🗖 🔀
<u>^</u>
(e)MMC_FLASH
Manufacturer ID: 0xFE
OEM/Application ID: 0x14E
Product name: MMCO2G
Product revision: 3.1
Manufacture date: Sep 2009

B::FLASHFILE.GETID

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>]</range></address></file>	Program the eMMC Flash.
FLASHFILE.LOAD <file> [<address>   <range>] /ComPare</range></address></file>	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 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):

<mark>職</mark> B::DATA.DUM	P VM:0x0 /DIALOG											
VM:0x0	Find Mo	dify La	ing 🔽	E	Track 🗹 Hex							
address	0	4 8	C	01234	56789ABCDEF							
VM:00000000	♦4E56533B 766552.	20 6F697369	75203A6E	;SVN_F	Revision:Lu 🔨							
VM:00000010	6F6E6B6E 0A0D6E	77 3B0A0D3B	43504D20	nknow	ութել։ թել։ "MPC 📑							
VM:00000020	58353535 43504D	2F 58363535	43504D2F	555X/I	MPC556X/MPC 🔤							
VM:00000030	58333535 504D20	2C 33363543	532F4D78	553X,	LMPC563xM/S 🔛							
VM:00000040	35354350 787840	33 6F4D282U	6F63616E	PC5631	MXXL(Monaco 🔥							
VM:00000050	40202029 363543	00 00203437	20203B0A	), LMP(			000					
VM:00000050	20202020 202020	20 20202020	20202020		B::FLASHFILE.D		000					$\sim$
VM-00000070	20202020 202020	20 20202020	2020202020		0.100000	LAAINI		ad Ma	difu	Laura At	Track M	20
VM:00000090	0A0D202D 6E4320	R 20206572	20202020	- CL+	000000	MAIN		IU		Long 🞽		10
VM:000000A0	20202020 326520	SA 0D203030	4D203B0A	- AREA	address	0	4		C	012345678	I9ABCDEF	_
VM:000000B0	66756E61 757463	61 20726572	46203A20	anufa	0000000000100000	+4E56533B	76655220	6F697369	75203A6E	;SVN_Rev1	sion:Lu	^
VM:000000C0	73656572 656061	53 3B0A0D20	3B0A0D20	reesca	000000000000000000000000000000000000000	6F6E686E	UAUD6E77	3BUAUD3B	43504020	nknown by;	SE; UMPC	
VM:000000D0	74754120 20726F	68 20202020	3A202020	LAuth	00000000000100020	58353535	4350402F	38363535	4350402F	SSSX/MPCS	S6X/MPC	~
VM:000000E0	49455220 3B0A0D	20 61745320	20737574	LREIL	000000000000000000000000000000000000000	26222222	2040202C	33363343 CE462020	03254070	DOSA, LMPC	(Monoco	
VM:000000F0	20202020 3A2020	20 6C657220	65736165		00000000000000000	40202029	26254250	00202020	20203804	) MDC567	d CL· -	^
VM:00000100	0A0D2064 2D2D20	3B 2D2D2D2D	2D2D2D2D	d_⊊⊧;.	000000000000000000000000000000000000000	20202020	20202020	20203437	2D203D0H	7, DHP COOT	TORF, O	
VM:00000110	20202020 202020	20 20202020	20202020		00000000000100070	2D2D2D2D	2D2D2D2D	2D2D2D2D	2D2D2D2D2D			
VI1:00000120 UM:00000120	20202020 202020	20 20202020	42504020		0000000000100080	2D2D2D2D	2D2D2D2D	2D2D2D2D	2D2D2D2D			
VM-00000130	59592525 222020	20 3000020	6204020	5577	0000000000100090	0A0D202D	6F43203B	20206572	20202020	-L&#;LCor	e	
VH.00000140	30303333 232040	0 0H0D2023	020H0D20	33882	00000000001000A0	20202020	3265203A	0D203030	4D203B0A	:_e20	ՍՍՏԷ:JM	
100	<u>S.</u>				000000000001000B0	66756E61	75746361	20726572	46203A20	anufactur	eruu:uF	
					000000000001000C0	73656572	656C6163	3B0A0D20	3B0A0D20	reescale	·RF; CF;	
					000000000001000D0	74754120	20726F68	20202020	3A202020	LAuthor		
					000000000001000E0	49455220	380A0D20	61745320	20737574	-REISSE:	Status_	
	Data is copie	d from th	e		000000000001000F0	20202020	3A202020	50557220	55735165	d CL.	.re lease	
-			-		000000000000000000000000000000000000000	20202054	20202038	20202020	2020202020	ULRF;L		
(	JPU to the el	MMC Fla	lsh	-	000000000000000000000000000000000000000	2020202020	2020202020	2020202020	2020202020			
					000000000000000000000000000000000000000	20202020	2020202020	3B0A0D20	43504D20		SET. MPC	
					00000000000100140	58583535	23284020	0A0D2029	620A0D20	55XX_@(#)	L'ELLELD	v
						<					SC-BE-	

## 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** 

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.D	UMP 0x100000	$\mathbf{X}$
0x100000	MAIN 💌 🎢 Find Modify 🛛 Long 💌 🗌 Trac	:k 🗹
addroce	0 4 8 C 0123456789ABCDEF	-
0000000000 100000	◆12345678 FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFF	~
000000000000000000000000000000000000000	TTTTTTTT FFFFFFF FFFFFFF FFFFFFF FFFFFFF	
0000000000100020	FFFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFF	. 😐
0000000000100030	FFFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFF	× .
0000000000100040	FFFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFF	
0000000000100050	FFFFFFFF FFFFFFF FFFFFFF FFFFFFF FFFFFF	-
	<	8

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.dum	p 0x1000						
0x1000	MAIN N	🖌 🦷 Fir	nd Moo	dify	Long 💌	Track	🕑 Hex
address	0	4	8		012343070	<b>JABCDEF</b>	
00000000000001000	♦E3510201	31510003	31A01201	31A02202		1815 <b>1</b> 81	~
00000000000001010	3AFFFFFA	E3510102	31510003	31A01081	AFF: SAQES	NQ193881	
00000000000001020	31A02082	<b>3AFFFFFA</b>	E3A00000	E1530001	2_815FF:N	N851N51	
00000000000001030	20433001	21800002	E15300A1	204330A1	10C_1888!1	NSEGOCL	~
00000000000001040	218000A2	E1530121	20433121	21800122	208!!\$S5!	1C_" ដូនូ !	~
00000000000001050	E15301A1	204331A1	218001A2	E3530000	ាត្តS§ឮ1CLg	18 NUSE	
00000000000001060	11B02222	11A01221	1AFFFFEF	E32C0000	""§1!101F	FFANN\S	
00000000000001070	42600000	E1A0F00E	E13C0000	42600000	NN^B5585N	8<588,0	
00000000000001080	E1A0F00E	33A00000	01A00FCC	03800001	858500835	SA SSNSE 10 HHUOX	
00000000000001090	E1A0F00E	E3510801	21A01821	23A02010	SEAE SEQE!	1811-8#	
0000000000000010A0	33A02000	E3510C01	21A01421	22822008	N-831FQ5	10 8 8" 40 5-2	
00000000000001080	E3510010	21A01221	22822004	E3510004	1 NQ5 128 4	42 FNQ5	
0000000000000010C0	82822003	908220A1	E35C0000	E1A00233	E 888 89N X-221-20U	N/83881	
0000000000000010D0	42600000	E1A0F00E	E52DE004	EB000230	NN BSFREE	5-50206	=
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU	E3A00000	E49DF004	E1A00000	E1A00000	UU03 TOD4U	0010001	
00000000000000000000000000000000000000	E1A00000	E1A00000	E1A00000	E1A00000	000100010	0010001	
0000000000001100	E3510000	UAUUUU32	42611000	E 1BUCUUU	0009520050	998.0222	
00000000000001110	42600000	E2512001	11500001	03A00000	NN BH-QEH	0010005	1.5
00000000000001120	81110002	00000002	9A000026	E3A02000	201120000	0040-05	
00000000000001130	E3510201	31510000	31A01201	32822004	ASVENNUTA	2015-22	
000000000000001140	3AFFFFFA	E3510102	31510000	31A01081	AFFICAR	0V13581	
000000000000001150	32822001	38555558	E2522003	BAUUUUUE	H-24AFF X	LK2000A	
00000000000001160	E1500001	20400001	E15000A1	20400041	HUPTHUR-1	OPTIOL.	
000000000000001170	E1500121	20400121	E 15001A1	204001A1	: HEJ : HOCA	AP11AQL	
	<						

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 0	N ; 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 O	FF				
SYStem.mode.attach					
wait 1.s					
if run()					
break					
GOSUB disable_watc	hdog				
D.S NSD:0x480B422C	: %LE %Long 0xe0f87 ; Configure the eMMC clock. ; LE = little endian				
<b>BREAK.RESet</b>					
<b>FLASHFILE.RESet</b>					
; Base address of <b>FLASHFILE.CONFIG 0</b>	the eMMC controller in the OMAP4430				

```
; 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> <file> <data range> 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
                                ; reset boundary scan configuration
BSDL.RESet
BSDL.FILE ./sam3u4e lqfp144.bsd ; load the required BSDL file
BSDL.HARDRESET
                                ; toggle TRST N pin
BSDL.SOFTRESET
                                 ; do a sequential JTAG reset
                                ; check, if BYPASS mode works
IF BSDL.CHECK.BYPASS()
(
  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 DATO to port PA5
   BSDL.FLASH.INIT SAFE
                                ; Initialize boundary scan chain to
                                 ; safe values according to
                                 : SAFE state from BSDL file
                                ; Enable flash programming
   FLASHFILE.BSDLaccess ON
                                 ; 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
```