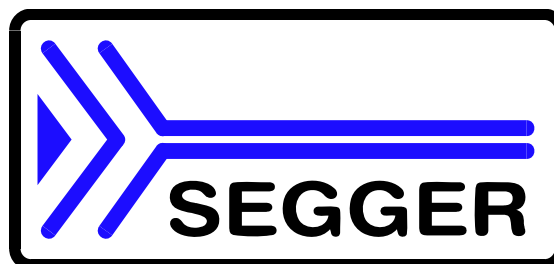


# ***J-Link***

## Reference manual for J-Link USB Protocol

Document RM08001-R7

Date: October 8, 2010



A product of SEGGER Microcontroller GmbH & Co. KG

[www.segger.com](http://www.segger.com)

## Disclaimer

Specifications written in this document are believed to be accurate, but are not guaranteed to be entirely free of error. The information in this manual is subject to change for functional or performance improvements without notice. Please make sure your manual is the latest edition. While the information herein is assumed to be accurate, SEGGER MICROCONTROLLER GmbH & Co. KG (the manufacturer) assumes no responsibility for any errors or omissions. The manufacturer makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. The manufacturer specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

## Copyright notice

You may not extract portions of this manual or modify the PDF file in any way without the prior written permission of the manufacturer. The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such a license.

© 2010 SEGGER Microcontroller GmbH & Co. KG, Hilden / Germany

## Trademarks

Names mentioned in this manual may be trademarks of their respective companies.

Brand and product names are trademarks or registered trademarks of their respective holders.

## Contact address

SEGGER Microcontroller GmbH & Co. KG

In den Weiden 11  
D-40721 Hilden

Germany

Tel. +49 2103-2878-0

Fax. +49 2103-2878-28

Email: support@segger.com

Internet: <http://www.segger.com>

## Document revisions

For further information on topics or routines not yet specified, please contact us.

| Revision | Date   | By | Explanation  |
|----------|--------|----|--|
| 0        | 080523 | OO | Initial version  |
| 1        | 080526 | OO | Chapter "Background information": Added JTAG buffer description.<br>Some minor changes.  |
| 2        | 080527 | OO | Some minor changes.  |
| 3        | 090520 | OO | Split chapter "USB communication" into chapter "USB" and chapter "Emulator protocol".<br>Some minor changes.                   |
| 4        | 090526 | AG | Chapter "Emulator protocol"<br>* Section "Protocol commands" updated.<br>* Section "Get system information functions" updated. |

| <b>Revision</b> | <b>Date</b> | <b>By</b> | <b>Explanation</b>   |
|-----------------|-------------|-----------|--|
| 5               | 090527      | AG        | Chapter "Emulator protocol"<br>* Section "Protocol commands" updated.                                    |
| 6               | 090616      | AG        | Chapter "Emulator protocol"<br>* Section "Protocol commands"<br>updated/corrected.                       |
| 7               | 101008      | OO        | Chapter "Emulator protocol"<br>* Added EMU_CMD_SELECT_IF<br>* Updated EMU_CMD_HW_JTAG3 for SWD<br>usage. |



# About this document

---

## Assumptions

This document assumes that you already have a solid knowledge of the following:

- The software tools used for building your application (assembler, linker, C compiler)
- The target processor.

## How to use this manual

This manual explains all the functions and macros that the product offers. It assumes you have a working knowledge of the C language. Knowledge of assembly programming is not required.

## Typographic conventions for syntax

This manual uses the following typographic conventions:

| Style      | Used for   |
|------------|--|
| Body       | Body text.   |
| Keyword    | Text that you enter at the command-prompt or that appears on the display (that is system functions, file- or pathnames). |
| Parameter  | Parameters in API functions.   |
| Sample     | Sample code in program examples.   |
| Reference  | Reference to chapters, sections, tables and figures or other documents.  |
| GUIElement | Buttons, dialog boxes, menu names, menu commands.  |
| Emphasis   | Very important sections  |

**Table 1.1: Typographic conventions**



# Table of Contents

|     |  |    |
|-----|--|----|
| 1   | Introduction .....                     | 9  |
| 1.1 | How to use this manual .....           | 10 |
| 1.2 | Purpose of this document .....         | 10 |
| 1.3 | Features of J-Link ARM .....           | 10 |
| 2   | Licensing .....                        | 11 |
| 2.1 | License terms .....                    | 12 |
| 3   | Background information .....           | 13 |
| 3.1 | JTAG scan chain values .....           | 14 |
| 3.2 | Emulator configuration .....           | 16 |
| 3.3 | JTAG data buffers .....                | 17 |
| 4   | USB .....                              | 19 |
| 4.1 | USB communication overview .....       | 20 |
| 4.2 | USB transactions .....                 | 21 |
| 4.3 | Using USBLib .....                     | 22 |
| 5   | Emulator protocol .....                | 23 |
| 5.1 | Communication protocol overview .....  | 24 |
| 5.2 | Protocol commands .....                | 25 |
| 5.3 | Get system information functions ..... | 27 |
| 5.4 | Get state information functions .....  | 35 |
| 5.5 | JTAG & Hardware functions .....        | 39 |
| 5.6 | Target functions .....                 | 57 |
| 5.7 | Configuration functions .....          | 66 |
| 6   | Glossary .....                         | 69 |
| 7   | Literature and references .....        | 73 |





# Chapter 1

## Introduction

---

This chapter describes the purpose of this manual and the J-Link ARM emulator which uses the ***J-Link USB Protocol***.

## 1.1 How to use this manual

This manual describes the J-Link USB Protocol used in J-Link products and how it is used.

In the course of this document the J-Link ARM is referred to as emulator.

## 1.2 Purpose of this document

The purpose of this document is to allow developers of open source software such as OpenOCD to write software which uses J-Link as debug interface.

This document gives an overview about the J-Link USB communication and how it can be used in your application. Sample communications to almost each command description should give you an idea of the data transferred via USB between J-Link and host.

## 1.3 Features of J-Link ARM



J-Link provides a feature rich list of functions to simplify the debugging process. All important functions which are used by the SEGGER software are useable via the USB protocol.

J-Link ARM has many features:

- USB 2.0 interface
- Any ARM7/ARM9 core supported, including thumb mode
- Automatic core recognition
- Maximum JTAG speed 12 MHz
- Download speed up to 720 Kbytes/second \*
- DCC speed up to 800 Kbytes/second \*
- No power supply required, powered through USB
- Auto speed recognition
- Support for adaptive clocking
- All JTAG signals can be monitored, target voltage can be measured
- Support for multiple devices
- Fully plug and play compatible
- A Standard 20-pin JTAG connector
- Optional 14-pin JTAG adapter available
- Wide target voltage range: 1.2V - 3.3V
- Optional adapter for 5V targets available

\* = Measured with J-Link Rev.5, ARM7 @ 50 MHz, 12MHz JTAG speed.

# Chapter 2

# Licensing

---

This chapter gives information about the licensing terms under which this document is published.

## 2.1 License terms

The information in this document may only be used when fully agreeing to the terms mentioned in this chapter.

Please note that the SEGGER J-Link software is copyrighted and may be used with original SEGGER J-Links and legal OEM versions only. For details, please refer to the License terms in `License.txt` which comes with the J-Link software and documentation pack.

In case of doubt, please contact us: [info@segger.com](mailto:info@segger.com) .

### 2.1.1 What you may do

You may use the information contained in this document to add support for the wide spread SEGGER J-Link ARM emulator and OEM versions in open source software.

### 2.1.2 What you are not allowed to do

You are not allowed to use the information in this document to

- build J-Link clones
- build J-Link compatible emulators.

# Chapter 3

## Background information

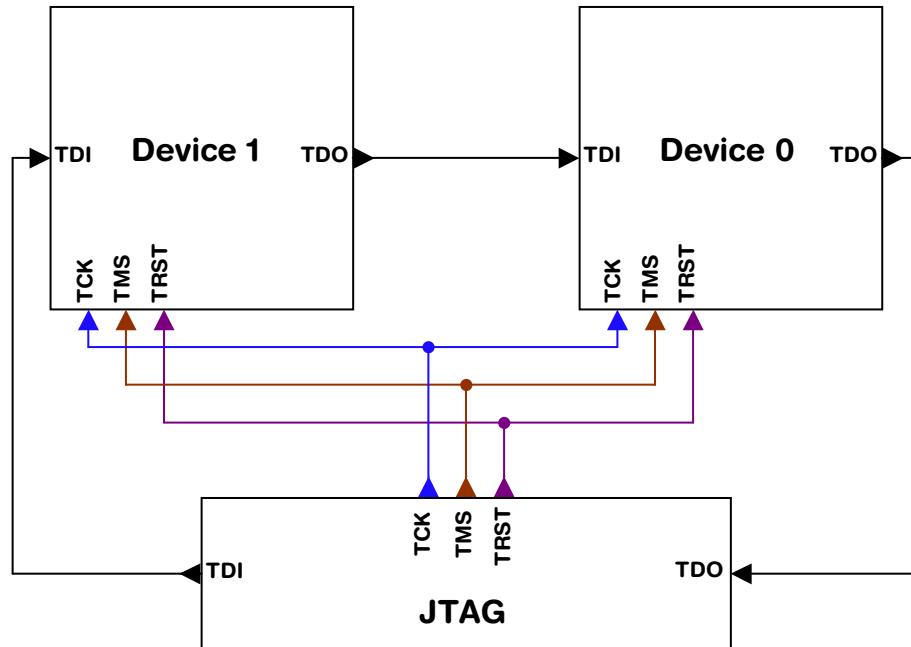
---

This chapter provides information required to understand parameters for certain USB commands when working with the J-Link emulator.

## 3.1 JTAG scan chain values

Some JTAG-related commands need an IRpre and DevicePos parameter. These values describe the position of a device in the scan chain and the shift count to reach that device.

The diagram below shows a scan chain configuration sample with 2 devices connected to the JTAG port.



### 3.1.1 IRPre

The IRPre value is the total number of bits in the instruction registers of the devices before the DUT counted from nearest device to TDO. The position can usually be seen in the schematic; the IR len can be found in the manual supplied by the manufacturers of the other devices.

ARM7 and ARM9 have an IR len of four.

For systems with just a single device, IRPre is 0.

For an example using the IRpre value please refer to the paragraph *Scan chain example* on page 15.

### 3.1.2 DevicePos

The DevicePos value of the DUT is the amount of devices in the scan chain before the DUT, counted from nearest device to TDO.

For systems with just a single device, DevicePos is 0.

For an example using the DevicePos value please refer to the paragraph *Scan chain example* on page 15.

### 3.1.3 Scan chain example

The following table shows a few sample configurations with 1,2 and 3 devices in different configurations. The DUT is marked in blue.

| Device 0<br>(IR len) | Device 1<br>(IR len) | Device 2<br>(IR len) | DevicePos | IR len<br>(IRpre) |
|----------------------|----------------------|----------------------|-----------|-------------------|
| ARM(4)               | -                    | -                    | 0         | 0                 |
| ARM(4)               | Xilinx(8)            | -                    | 0         | 0                 |
| Xilinx(8)            | ARM(4)               | -                    | 1         | 8                 |
| Xilinx(8)            | Xilinx(8)            | ARM(4)               | 2         | 16                |
| ARM(4)               | Xilinx(8)            | ARM(4)               | 0         | 0                 |
| ARM(4)               | Xilinx(8)            | ARM(4)               | 2         | 12                |
| Xilinx(8)            | ARM(4)               | Xilinx(8)            | 1         | 8                 |

**Table 3.1: Example scan chain**

## 3.2 Emulator configuration

Emulator configuration is a 256 byte non-volatile memory area which can be read and written via USB. It holds the emulator configuration that was setup for your J-Link. The following table shows the current configuration offsets and their meanings.

| Offset      | Values   | Explanation   |
|-------------|--|---|
| 0x00        | 0x00 - 0x03 : USB-Address 0-3<br>0xFF : Default                              | USB-Address of the emulator. Allows usage of more than one J-Link at the same time. |
| 0x01 - 0x03 | 0xFF   | Reserved  |
| 0x04 - 0x07 | 0xFFFFFFFF : Default, on if J-Link KS<br>0x00000000 : Off<br>0x00000001 : On | Kickstart power on JTAG-pin 19.   |
| 0x08 - 0x1F | 0xFF   | Reserved  |
| 0x20 - 0x23 | IP-Address in hex format.<br>0xFFFFFFFF if not configured.                   | IP-Address (only for J-Link Pro).   |
| 0x24 - 0x27 | Subnet mask in hex format.<br>0xFFFFFFFF if not configured.                  | Subnetmask (only for J-Link Pro).   |
| 0x28 - 0x2F | 0xFF   | Reserved  |
| 0x30 - 0x35 | MAC-Address<br>0xFFFFFFFFFFFFFFF if not configured.                          | MAC-Address (only for J-Link Pro).  |
| 0x36 - 0xFF | 0xFF   | Reserved  |

**Table 3.2: Emulator configuration overview**



### 3.3 JTAG data buffers

The J-Link has three JTAG data buffers. Two of these are output buffers used for TMS and TDI, the third is an input buffer for TDO data. To work with the J-Link, an understanding of the buffers and the way the data is stored in them is quite useful.

#### 3.3.1 Explanation of terms

In this document input and output buffers are seen from host perspective.

##### Input buffer

The input buffer stores the incoming TDO signals from the device.

##### Output buffer

Output buffers stores TMS and TDI signals which are transferred to the device.

#### 3.3.2 Organization of buffers

Model of JTAG Buffer:

|       |     |     |     |     |     |     |     |     |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| byte0 | b7  | ... | ... | ... | ... | ... | ... | b0  |
| byte1 | b15 | ... | ... | ... | ... | ... | ... | b8  |
| byte2 | b23 | ... | ... | ... | ... | ... | ... | b16 |
| byte3 | b31 | ... | ... | ... | ... | ... | ... | b24 |
| byte4 | b39 | ... | ... | ... | ... | ... | ... | b32 |
| byte5 | b47 | ... | ... | ... | ... | ... | ... | b40 |
| byte6 | b55 | ... | ... | ... | ... | ... | ... | b48 |
| ...   | ... | ... | ... | ... | ... | ... | ... | ... |

All three JTAG data buffers are organized the same way:

Bit  $n$  of the bit stream is stored in byte  $n/8$ , bit  $n\%8$ . This means the first byte is sent first, least significant bit (lsb) first.

The same thing is true for data received: The first bit received is the lsb of byte 0.

##### Size of buffers

All buffers are big enough to hold 2 KByte of data.



# Chapter 4

## USB

---

This chapter explains the USB communication and specifics which have to be taken care of when using USB communication under Windows and other operating systems.

## 4.1 USB communication overview

J-Link uses USB bulk communication to transfer data between host and J-Link. Various specifics which are mandatory are explained in this section.

### 4.1.1 USB endpoints

Newer J-Links use 2 bulk endpoints, one for "IN" (EP1) and one for "OUT" (EP2) communication whereas old J-Links (V3, V4) only use one endpoint (EP1) for "IN" and "OUT" communication. Data direction is seen from host side so "IN" means from the J-Link to the host.

The following table gives a quick overview which hardware version uses what USB endpoints:

| Hardware version                                      | Endpoint used for "IN" | Endpoint used for "OUT" |
|---|------------------------|-------------------------|
| J-Link V3 / J-Link V4                                 | 1                      | 1                       |
| J-Link V5 and versions above                          | 1                      | 2                       |
| J-Link compatible hardware (Flasher ARM, J-Trace,...) | 1                      | 2                       |

**Table 4.1: USB endpoint overview**

### 4.1.2 Using different operating systems

Depending on the operating system being used, there are different ways to communicate with the J-Link via USB.

It is possible to use J-Link under almost every operating system as long as there is a USB bulk driver which allows communication with the J-Link.

#### Using a Windows operating system

Using Windows for communication with the J-Link you will need to have a USB bulk kernel mode driver as Windows does not come with such a feature aboard. This driver, named the J-Link USB driver, is shipped together with the Windows version of the J-Link software package.

#### Using a non Windows operating system

Using an operating system other than Windows you will have to find an other way to operate the J-Link via USB. Under different operating systems there is a freeware package called "USBLib" available which allows you to communicate with the J-Link.

### 4.1.3 Endianness

All data units larger than a single byte are transferred little endian, meaning least significant bytes are transferred first.

#### U16 Example

U16 data to send:

0xCDEF

Data sent via USB:

EF CD

#### U32 Example

U32 data to send:

0x12345678

Data sent via USB:

78 56 34 12

## 4.2 USB transactions

If multiple bytes are send to the emulator or send by the emulator, there is a choice: The information can be send in a single transaction or in multiple transactions. So both sides are free to send data in one or more transactions. However, due to the long USB latencies, it is recommended to use as few transactions as possible.

### 4.2.1 Timeout

J-Link uses a 5 second timeout for all USB operations.

This means that every command needs to be completed within 5 seconds. If 5 seconds is too short (for example for a long JTAG sequence at a low speed) the JTAG sequence needs to be split into 2 or more shorter sequences.

### 4.2.2 USB latency

The two most important limiting factors in communication between the host and the emulator are:

- USB Transmission speed
- USB latency

The transmission speed is fixed to 12MHz and allows transmission of up to app. 1MByte/sec. This is sufficient for fast operation of the emulator.

The USB latency is more of a problem. Since the host schedules USB transactions in frames of 1ms for full speed and 125us for high speed connections, the average latency is 2ms for full speed and 250us for high speed connections. This means that even a small transaction of 1byte typically takes 2ms on full speed connection and it also means that minimizing the number of transactions to the minimum required is desirable.

### 4.2.3 OUT transaction (downstream)

In an OUT-transaction, the host sends data to the emulator.

This is a pretty straightforward process; if the host wants to send a number of bytes to the J-Link, it can simply start such a write transaction.

The maximum size of one transaction is 0xFFFF bytes.

### 4.2.4 IN-transaction

In an IN-transaction, the host requests data from the emulator. This is a is a more tricky process.

The maximum size of one transaction is 0x8000 bytes.

A transaction is followed by a 0-byte packet if the transaction size is a multiple of 64 bytes, but not the maximum packet size.

## 4.3 Using USBLib

USBLib is freeware and can be used on most platforms.

Again, downstream (OUT-) transactions are unproblematic.

For upstream (IN-) transactions, transaction termination needs to be handled correctly.

One easy way of doing so is to not request multiples of 64 bytes.

So if 256 bytes need to be read from the J-Link, instead 255 bytes, then 1 byte could be read.

# Chapter 5

## Emulator protocol

---

This chapter explains the communication protocol used by the J-Link emulator to communicate with the host.

## 5.1 Communication protocol overview

The J-Link firmware uses several commands to communicate with the host software.

The communication is always initiated by the host. The host sends an `U8` command to the emulator followed by parameters being known to be needed by this command.

The firmware retrieves the first byte from the buffer, containing the command and executes the proper functions. The parameters expected are read from the communication buffer by the subroutines that are called through the command. Return values are sent back via USB.



## 5.2 Protocol commands

The list below gives an overview about the usable commands:

| Define                               | Value | Explanation  |
|--------------------------------------|-------|--|
| Get system information functions     |       |  |
| EMU_CMD_VERSION                      | 0x01  | Retrieves the firmware version.  |
| EMU_CMD_GET_SPEEDS                   | 0xC0  | Retrieves the base freq. and the min. divider of the emulator CPU.             |
| EMU_CMD_GET_MAX_MEM_BLOCK            | 0xD4  | Retrieves the maximum memory block-size.                                       |
| EMU_CMD_GET_CAPS                     | 0xE8  | Retrieves capabilities of the emulator.  |
| EMU_CMD_GET_CAPS_EX                  | 0xED  | Retrieves capabilities (including extended ones) of the emulator.              |
| EMU_CMD_GET_HW_VERSION               | 0xF0  | Retrieves the hardware version of the emulator.                                |
| Get state information functions      |       |  |
| EMU_CMD_GET_STATE                    | 0x07  | Retrieves voltage and JTAG pin states.   |
| EMU_CMD_GET_HW_INFO                  | 0xC1  | Retrieves hardware information.  |
| EMU_CMD_GET_COUNTERS                 | 0xC2  | Retrieves target connection counters.  |
| EMU_CMD_MEASURE_RTCK_REACT           | 0xF6  | Measures RTCK reaction time.   |
| JTAG & Hardware functions            |       |  |
| EMU_CMD_RESET_TRST                   | 0x02  | Resets TRST.   |
| EMU_CMD_SET_SPEED                    | 0x05  | Sets the JTAG speed.   |
| EMU_CMD_SELECT_IF                    | 0xC7  | Selects the target interface or retrieves current configured target interface. |
| EMU_CMD_SET_KS_POWER                 | 0x08  | Sets KS power to on or off.  |
| EMU_CMD_HW_CLOCK                     | 0xC8  | Generates one clock and retrieves TDI.   |
| EMU_CMD_HW_TMS0                      | 0xC9  | Clears TMS.  |
| EMU_CMD_HW_TMS1                      | 0xCA  | Sets TMS.  |
| EMU_CMD_HW_DATA0                     | 0xCB  | Clears TDI.  |
| EMU_CMD_HW_DATA1                     | 0xCC  | Sets TDI.  |
| EMU_CMD_HW_JTAG                      | 0xCD  | Handles JTAG connection.   |
| EMU_CMD_HW_JTAG2                     | 0xCE  | Handles JTAG connection with word aligned data transfers.                      |
| EMU_CMD_HW_JTAG3                     | 0xCF  | Handles JTAG connection with word aligned data transfers and return value.     |
| EMU_CMD_HW_JTAG_WRITE                | 0xD5  | Same as EMU_CMD_HW_JTAG3 but without response data on TDO.                     |
| EMU_CMD_HW_JTAG_GET_RESULT           | 0xD6  | Receive status of sticky error bit which is handled by EMU_CMD_HW_JTAG_WRITE.  |
| EMU_CMD_HW_TRST0                     | 0xDE  | Activates TRST.  |
| EMU_CMD_HW_TRST1                     | 0xDF  | Deactivates TRST.  |
| EMU_CMD_WRITE_DCC                    | 0xF1  | Writes to JTAG using DCC.  |
| Target functions                     |       |  |
| EMU_CMD_RESET_TARGET                 | 0x03  | Resets the target.   |
| EMU_CMD_HW_RELEASE_RESET_ST_OP_EX    | 0xD0  | Resets the CPU and halts as soon as its possible or the timeout expires.       |
| EMU_CMD_HW_RELEASE_RESET_ST_OP_TIMED | 0xD1  | Resets the CPU and halts as soon as its possible or the timeout expires.       |
| EMU_CMD_HW_RESET0                    | 0xDC  | Activates target reset.  |

**Table 5.1: Protocol command overview**

| Define                  | Value | Explanation                                  |
|-------------------------|-------|--|
| EMU_CMD_HW_RESET1       | 0xDD  | Deactivates target reset.                    |
| EMU_CMD_GET_CPU_CAPS    | 0xE9  | Retrieves capabilities of the CPU.           |
| EMU_CMD_EXEC_CPU_CMD    | 0xEA  | Executes a CPU command.                      |
| EMU_CMD_WRITE_MEM       | 0xF4  | Writes to target memory.                     |
| EMU_CMD_READ_MEM        | 0xF5  | Reads from target memory.                    |
| EMU_CMD_WRITE_MEM_ARM79 | 0xF7  | Writes to target memory on ARM 7/9 targets.  |
| EMU_CMD_READ_MEM_ARM79  | 0xF8  | Reads from target memory on ARM 7/9 targets. |
| Configuration functions |       |  |
| EMU_CMD_READ_CONFIG     | 0xF2  | Reads the emulator configuration.            |
| EMU_CMD_WRITE_CONFIG    | 0xF3  | Writes the emulator configuration.           |

**Table 5.1: Protocol command overview**

## 5.3 Get system information functions

These commands are used to retrieve information about the firmware used by the emulator.

### 5.3.1 EMU\_CMD\_VERSION

#### Description

Retrieves the length and content of the firmware version string of the emulator.

#### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0x01 |

Table 5.2: EMU\_CMD\_VERSION command overview

#### Response

| Direction | Data               | Name     | Explanation  |
|-----------|--------------------|----------|--|
| H<-E      | 1 * U16            | NumBytes | Length of the firmware string in bytes. Typically 0x70 . |
| H<-E      | <NumBytes><br>* U8 | Version  | Firmware version string.                                 |

Table 5.3: EMU\_CMD\_VERSION response overview

#### Additional information

This is typically the first command sent.

#### Sample communication

Retrieve firmware string command.

H->E: 01

Returns the firmware string length.

H<-E: 70

Returns the firmware string.

```
H<-E: 4A 2D 4C 69 6E 6B 20 63 6F 6D 70 69 6C 65 64 20 J-Link compiled
      44 65 63 20 30 33 20 32 30 30 37 20 31 37 3A 31 Dec 03 2007 17:1
      35 3A 33 31 20 41 52 4D 20 52 65 76 2E 35 00 00 5:31 ARM Rev.5..
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

## 5.3.2 EMU\_CMD\_GET\_SPEEDS

### Description

Retrieves the base freq. and the min. divider of the emulator CPU.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xC0 |

**Table 5.4: EMU\_CMD\_GET\_SPEEDS command overview**

### Response

| Direction | Data    | Name     | Explanation                                   |
|-----------|---------|----------|---|
| H<-E      | 1 * U32 | BaseFreq | Retrieves the base frequency of emulator CPU. |
| H<-E      | 1 * U16 | MinDiv   | Retrieves the min. divider of emulator CPU.   |

**Table 5.5: EMU\_CMD\_GET\_SPEEDS response overview**

### Sample communication

Retrieve base freq. and min. divider of emulator CPU.

H->E: C0

Returns 48MHz base freq. with min. divider of 4.

H<-E: 00 6C DC 02 04 00

### 5.3.3 EMU\_CMD\_GET\_MAX\_MEM\_BLOCK

#### Description

Retrieves the maximum memory blocksize.

#### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xD4 |

**Table 5.6: EMU\_CMD\_GET\_MAX\_MEM\_BLOCK command overview**

#### Response

| Direction | Data    | Name    | Explanation                  |
|-----------|---------|---------|------------------------------|
| H<-E      | 1 * U32 | MaxSize | Returned maximum block size. |

**Table 5.7: EMU\_CMD\_GET\_MAX\_MEM\_BLOCK response overview**

## 5.3.4 EMU\_CMD\_GET\_CAPS

### Description

Retrieves capabilities of the emulator.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xE8 |

Table 5.8: EMU\_CMD\_GET\_CAPS command overview

### Response

| Direction | Data    | Name | Explanation                             |
|-----------|---------|------|---|
| H<-E      | 1 * U32 | Caps | Capabilities supported by the emulator. |

Table 5.9: EMU\_CMD\_GET\_CAPS response overview

### Additional information

The capability flags used are described in the table below:

| Bit | Define                     | Explanation   |
|-----|----------------------------|---|
| 0   | EMU_CAP_RESERVED           | Always 1.   |
| 1   | EMU_CAP_GET_HW_VERSION     | Supports command "EMU_CMD_GET_HARDWARE_VERSION"         |
| 2   | EMU_CAP_WRITE_DCC          | Supports command "EMU_CMD_WRITE_DCC"                    |
| 3   | EMU_CAP_ADAPTIVE_CLOCKING  | Supports adaptive clocking                              |
| 4   | EMU_CAP_READ_CONFIG        | Supports command "EMU_CMD_READ_CONFIG"                  |
| 5   | EMU_CAP_WRITE_CONFIG       | Supports command "EMU_CMD_WRITE_CONFIG"                 |
| 6   | EMU_CAP_TRACE              | Supports trace commands                                 |
| 7   | EMU_CAP_WRITE_MEM          | Supports command "EMU_CMD_WRITE_MEM"                    |
| 8   | EMU_CAP_READ_MEM           | Supports command "EMU_CMD_READ_MEM"                     |
| 9   | EMU_CAP_SPEED_INFO         | Supports command "EMU_CMD_GET_SPEED"                    |
| 10  | EMU_CAP_EXEC_CODE          | Supports commands "EMU_CMD_CODE_..."                    |
| 11  | EMU_CAP_GET_MAX_BLOCK_SIZE | Supports command "EMU_CMD_GET_MAX_BLOCK_SIZE"           |
| 12  | EMU_CAP_GET_HW_INFO        | Supports commands "EMU_CMD_GET_HW_INFO"                 |
| 13  | EMU_CAP_SET_KS_POWER       | Supports commands "EMU_CMD_SET_KS_POWER"                |
| 14  | EMU_CAP_RESET_STOP_TIMED   | Supports commands "EMU_CMD_HW_RELEASE_RESET_STOP_TIMED" |
| 15  | -                          | Reserved  |
| 16  | EMU_CAP_MEASURE_RTCK_REACT | Supports commands "EMU_CMD_MEASURE_RTCK_REACT"          |
| 17  | EMU_CAP_SELECT_IF          | Supports command "EMU_CMD_HW_SELECT_IF"                 |

Table 5.10: EMU\_CMD\_GET\_CAPS flag overview

| Bit | Define                                     | Explanation  |
|-----|--|--|
| 18  | <a href="#">EMU_CAP_RW_MEM_ARM79</a>       | Supports command "EMU_CMD_WRITE_MEM_ARM79", "EMU_CMD_READ_MEM_ARM79" |
| 19  | <a href="#">EMU_CAP_GET_COUNTERS</a>       | Supports command "EMU_CMD_GET_COUNTERS"                              |
| 20  | <a href="#">EMU_CAP_READ_DCC</a>           | Supports command "EMU_CMD_READ_DCC"                                  |
| 21  | <a href="#">EMU_CAP_GET_CPU_CAPS</a>       | Supports command "EMU_CMD_GET_CPU_CAPS"                              |
| 22  | <a href="#">EMU_CAP_EXEC_CPU_CMD</a>       | Supports command "EMU_CMD_EXEC_CPU_CMD"                              |
| 23  | <a href="#">EMU_CAP_SWO</a>                | Supports command "EMU_CMD_SWO"                                       |
| 24  | <a href="#">EMU_CAP_WRITE_DCC_EX</a>       | Supports command "EMU_CMD_WRITE_DCC_EX"                              |
| 25  | <a href="#">EMU_CAP_UPDATE_FIRMWARE_EX</a> | Supports command "EMU_CMD_UPDATE_FIRMWARE_EX"                        |
| 26  | <a href="#">EMU_CAP_FILE_IO</a>            | Supports command "EMU_CMD_FILE_IO"                                   |
| 27  | <a href="#">EMU_CAP_REGISTER</a>           | Supports command "EMU_CMD_REGISTER"                                  |
| 28  | <a href="#">EMU_CAP_INDICATORS</a>         | Supports command "EMU_CMD_INDICATORS"                                |
| 29  | <a href="#">EMU_CAP_TEST_NET_SPEED</a>     | Supports command "EMU_CMD_TEST_NET_SPEED"                            |
| 30  | <a href="#">EMU_CAP_RAWTRACE</a>           | Supports command "EMU_CMD_RAWTRACE"                                  |
| 31  | -  | Reserved   |

**Table 5.10: EMU\_CMD\_GET\_CAPS flag overview**

### Sample communication

Retrieve emulator capabilities.

H->E: E8

Returns all capabilities except trace commands.

H<-E: BF 7F FF 3B

## 5.3.5 EMU\_CMD\_GET\_CAPS\_EX

### Description

Retrieves extended capabilities of the emulator.

## Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xED |

Table 5.11: EMU\_CMD\_GET\_CAPS\_EX command overview

## Response

| Direction | Data    | Name | Explanation                             |
|-----------|---------|------|---|
| H<-E      | 32 * U8 | Caps | Capabilities supported by the emulator. |

Table 5.12: EMU\_CMD\_GET\_CAPS\_EX response overview

## Additional information

The capability flags used are described in the table below:

| Bit | Define                        | Explanation  |
|-----|-------------------------------|--|
| 0   | EMU_CAP_EX_RESERVED           | Always 1.  |
| 1   | EMU_CAP_EX_GET_HW_VERSION     | Supports command "EMU_CMD_GET_HARDWARE_VERSION"                      |
| 2   | EMU_CAP_EX_WRITE_DCC          | Supports command "EMU_CMD_WRITE_DCC"                                 |
| 3   | EMU_CAP_EX_ADAPTIVE_CLOCKING  | Supports adaptive clocking   |
| 4   | EMU_CAP_EX_READ_CONFIG        | Supports command "EMU_CMD_READ_CONFIG"                               |
| 5   | EMU_CAP_EX_WRITE_CONFIG       | Supports command "EMU_CMD_WRITE_CONFIG"                              |
| 6   | EMU_CAP_EX_TRACE              | Supports trace commands  |
| 7   | EMU_CAP_EX_WRITE_MEM          | Supports command "EMU_CMD_WRITE_MEM"                                 |
| 8   | EMU_CAP_EX_READ_MEM           | Supports command "EMU_CMD_READ_MEM"                                  |
| 9   | EMU_CAP_EX_SPEED_INFO         | Supports command "EMU_CMD_GET_SPEED"                                 |
| 10  | EMU_CAP_EX_EXEC_CODE          | Supports commands "EMU_CMD_CODE_..."                                 |
| 11  | EMU_CAP_EX_GET_MAX_BLOCK_SIZE | Supports command "EMU_CMD_GET_MAX_BLOCK_SIZE"                        |
| 12  | EMU_CAP_EX_GET_HW_INFO        | Supports commands "EMU_CMD_GET_HW_INFO"                              |
| 13  | EMU_CAP_EX_SET_KS_POWER       | Supports commands "EMU_CMD_SET_KS_POWER"                             |
| 14  | EMU_CAP_EX_RESET_STOP_TIME    | Supports commands "EMU_CMD_HW_RELEASE_RESET_STOP_TIME"               |
| 15  | -                             | Reserved   |
| 16  | EMU_CAP_EX_MEASURE_RTCK_REACT | Supports commands "EMU_CMD_MEASURE_RTCK_REACT"                       |
| 17  | EMU_CAP_EX_SELECT_IF          | Supports command "EMU_CMD_HW_SELECT_IF"                              |
| 18  | EMU_CAP_EX_RW_MEM_ARM79       | Supports command "EMU_CMD_WRITE_MEM_ARM79", "EMU_CMD_READ_MEM_ARM79" |
| 19  | EMU_CAP_EX_GET_COUNTERS       | Supports command "EMU_CMD_GET_COUNTERS"                              |
| 20  | EMU_CAP_EX_READ_DCC           | Supports command "EMU_CMD_READ_DCC"                                  |

Table 5.13: EMU\_CMD\_GET\_CAPS\_EX flag overview



| Bit | Define  | Explanation                                   |
|-----|---|---|
| 21  | <a href="#">EMU_CAP_EX_GET_CPU_CAPS</a>       | Supports command "EMU_CMD_GET_CPU_CAPS"       |
| 22  | <a href="#">EMU_CAP_EX_EXEC_CPU_CMD</a>       | Supports command "EMU_CMD_EXEC_CPU_CMD"       |
| 23  | <a href="#">EMU_CAP_EX_SWO</a>                | Supports command "EMU_CMD_SWO"                |
| 24  | <a href="#">EMU_CAP_EX_WRITE_DCC_EX</a>       | Supports command "EMU_CMD_WRITE_DCC_EX"       |
| 25  | <a href="#">EMU_CAP_EX_UPDATE_FIRMWARE_EX</a> | Supports command "EMU_CMD_UPDATE_FIRMWARE_EX" |
| 26  | <a href="#">EMU_CAP_EX_FILE_IO</a>            | Supports command "EMU_CMD_FILE_IO"            |
| 27  | <a href="#">EMU_CAP_EX_REGISTER</a>           | Supports command "EMU_CMD_REGISTER"           |
| 28  | <a href="#">EMU_CAP_EX_INDICATORS</a>         | Supports command "EMU_CMD_INDICATORS"         |
| 29  | <a href="#">EMU_CAP_EX_TEST_NET_SPEED</a>     | Supports command "EMU_CMD_TEST_NET_SPEED"     |
| 30  | <a href="#">EMU_CAP_EX_RAWTRACE</a>           | Supports command "EMU_CMD_RAWTRACE"           |
| 31  | <a href="#">EMU_CAP_EX_GET_CAPS_EX</a>        | Supports command "EMU_CMD_GET_CAPS_EX"        |
| 32  | <a href="#">EMU_CAP_EX_HW_JTAG_WRITE</a>      | Supports command "EMU_CMD_HW_JTAG_WRITE"      |

**Table 5.13: EMU\_CMD\_GET\_CAPS\_EX flag overview**

### Sample communication

Retrieve emulator capabilities.

H->E: ED

Returns all capabilities except trace commands.

```
H<-E: BF 7F FF BB 01 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

## 5.3.6 EMU\_CMD\_GET\_HW\_VERSION

### Description

Retrieves the product hardware version of the emulator.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xF0 |

Table 5.14: EMU\_CMD\_GET\_HW\_VERSION command overview

### Response

| Direction | Data    | Name    | Explanation                       |
|-----------|---------|---------|-----------------------------------|
| H<-E      | 1 * U32 | Version | Hardware version of the emulator. |

Table 5.15: EMU\_CMD\_GET\_HW\_VERSION response overview

### Additional information

The product hardware version is returned in the following format.

TTMMmmrr

This stands for:

TT: Hardware type

MM: Major version

mm: Minor version

rr: Revision

Example: The version 0052000 stands for a J-Link Version 5.2 .

If the capability EMU\_CMD\_GET\_HW\_VERSION is not enabled `Version` is always 0x00 .

### Sample communication

Retrieve hardware version command.

H->E: F0

Returns the hardware version 0060000 which equals a J-Link V6.

H<-E: 60 EA 00 00

### J-Link hardware versions

The following table gives an overview about what hardware versions can be retrieved using this command:

| Hardware type | Name       |
|---------------|------------|
| 0             | J-Link     |
| 1             | J-Trace    |
| 2             | Flasher    |
| 3             | J-Link Pro |

## 5.4 Get state information functions

These commands are used to retrieve information about the actual status of the emulator.

### 5.4.1 EMU\_CMD\_GET\_STATE

#### Description

Retrieves the voltage of the target and the JTAG pin states for TCK, TDI, TDO, TMS, TRES and TRST.

#### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0x07 |

Table 5.16: EMU\_CMD\_GET\_STATE command overview

#### Response

| Direction | Data    | Name    | Explanation           |
|-----------|---------|---------|-----------------------|
| H<-E      | 1 * U16 | Voltage | VCC stored in unit mV |
| H<-E      | 1 * U8  | TCK     | JTAG pin TCK state.   |
| H<-E      | 1 * U8  | TDI     | JTAG pin TDI state.   |
| H<-E      | 1 * U8  | TDO     | JTAG pin TDO state.   |
| H<-E      | 1 * U8  | TMS     | JTAG pin TMS state.   |
| H<-E      | 1 * U8  | TRES    | JTAG pin TRES state.  |
| H<-E      | 1 * U8  | TRST    | JTAG pin TRST state.  |

Table 5.17: EMU\_CMD\_GET\_STATE response overview

#### Sample communication

Retrieve state information command.

H->E: 07

Returns 3.267V; TCK, TRES and TRST are high; TDI, TDO, TMS are low.

H<-E: C3 0C 01 00 00 00 01 01

## 5.4.2 EMU\_CMD\_GET\_HW\_INFO

### Description

Retrieves information about the KS power of the emulator.

### Command

| Direction | Data    | Name | Explanation   |
|-----------|---------|------|---|
| H->E      | 1 * U8  | Cmd  | Command: 0xC1   |
| H->E      | 1 * U32 | Mask | Bitmask selecting the information that should be retrieved. |

Table 5.18: EMU\_CMD\_GET\_HW\_INFO command overview

### Response

| Direction | Data    | Name    | Explanation  |
|-----------|---------|---------|--|
| H<-E      | x * U32 | HW Info | Hardware information returned. x means the amount of hardware information defined by the <a href="#">Mask</a> flags. |

Table 5.19: EMU\_CMD\_GET\_HW\_INFO response overview

### Additional information

The flags that can be used as [Mask](#) are described in the table below:

| Bit | Define   | Explanation   |
|-----|--|---|
| 0   | <a href="#">HW_INFO_POWER_ENABLED</a>          | Retrieves KS power status.<br>0x00000000: Power is off<br>0x00000001: Power is on   |
| 1   | <a href="#">HW_INFO_POWER_OVERCURRENT</a>      | Retrieves information about why the target power was switched off.<br>0x00000000: Everything is normal<br>0x00000001: 2ms @ 3000mA<br>0x00000002: 10ms @ 1000mA<br>0x00000003: 40ms @ 400mA |
| 2   | <a href="#">HW_INFO_ITARGET</a>                | Consumption of the target in mA.  |
| 3   | <a href="#">HW_INFO_ITARGET_PEAK</a>           | Peak consumption of the target in mA.   |
| 4   | <a href="#">HW_INFO_ITARGET_PEAK_OPERATION</a> | Peak consumption of the target in mA while operating. In most cases this is the same value as <a href="#">HW_INFO_ITARGET_PEAK</a> .  |
| 10  | <a href="#">HW_INFO_ITARGET_MAX_TIME0</a>      | Max. time in ms the consumption of the target exceeded<br><a href="#">HW_INFO_POWER_OVERCURRENT</a> type 0x01.  |
| 11  | <a href="#">HW_INFO_ITARGET_MAX_TIME1</a>      | Max. time in ms the consumption of the target exceeded<br><a href="#">HW_INFO_POWER_OVERCURRENT</a> type 0x02.  |
| 12  | <a href="#">HW_INFO_ITARGET_MAX_TIME2</a>      | Max. time in ms the consumption of the target exceeded<br><a href="#">HW_INFO_POWER_OVERCURRENT</a> type 0x03.  |

Table 5.20: EMU\_CMD\_GET\_HW\_INFO flag overview

The amount of U32 data returned by this command depends on how many flags you have used in your [Mask](#). Undefined flags return 0xFFFFFFFF.

### Sample communication

Retrieve hardware information about KS power status and target consumption.

H->E: C1 05 00 00 00

Returns KS power is on and the target consumes 12mA.

H<-E: 01 00 00 00 0C 00 00 00

## 5.4.3 EMU\_CMD\_GET\_COUNTERS

### Description

Retrieves counter values of how long a target is connected and how many times a target was connected / disconnected.

### Command

| Direction | Data    | Name | Explanation   |
|-----------|---------|------|---|
| H->E      | 1 * U8  | Cmd  | Command: 0xC2   |
| H->E      | 1 * U32 | Mask | Bitmask selecting the information that should be retrieved. |

**Table 5.21: EMU\_CMD\_GET\_COUNTERS command overview**

### Response

| Direction | Data    | Name     | Explanation  |
|-----------|---------|----------|--|
| H<-E      | x * U32 | Counters | Returned counter values. x means the amount of counter values defined by the <a href="#">Mask</a> flags. |

**Table 5.22: EMU\_CMD\_GET\_COUNTERS response overview**

### Additional information

The flags that can be used as [Mask](#) are described in the table below:

| Bit | Define                                 | Explanation   |
|-----|--|---|
| 0   | <a href="#">CNT_INDEX_POWER_ON</a>     | Retrieves the counter describing how many ms a powered target is connected. The counter is reset after 24h. |
| 1   | <a href="#">CNT_INDEX_POWER_CHANGE</a> | Retrieves the counter describing how many times a powered target was connected or disconnected.             |

**Table 5.23: EMU\_CMD\_GET\_COUNTERS flag overview**

The amount of U32 data returned by this command depends on how many flags you have used in your [Mask](#). Undefined flags return 0xFFFFFFFF.

### Sample communication

Requesting value of the power change counter.

H->E: C2 02 00 00 00

Returns 5 power changes.

H<-E: 05 00 00 00

## 5.4.4 EMU\_CMD\_MEASURE\_RTCK\_REACT

### Description

Measures the RTCK reaction time of the target device.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xF6 |

**Table 5.24: EMU\_CMD\_MEASURE\_RTCK\_REACT command overview**

### Response

| Direction | Data    | Name    | Explanation   |
|-----------|---------|---------|---|
| H<-E      | 1 * U32 | Result  | 0x00000000: O.K.<br>0x00000001: RTCK did not react on time. |
| H<-E      | 1 * U32 | Minimum | Minimum RTCK reaction time in ns.                           |
| H<-E      | 1 * U32 | Maximum | Maximum RTCK reaction time in ns.                           |
| H<-E      | 1 * U32 | Average | Average RTCK reaction time in ns.                           |

**Table 5.25: EMU\_CMD\_MEASURE\_RTCK\_REACT response overview**

### Sample communication

Measure RTCK reaction time command.

H->E: F6

Returns O.K., Min. 0ns, Max. 126ns and Average 25ns reaction time.

H<-E: 00 00 00 00 00 00 00 00 7E 00 00 00 19 00 00 00

## 5.5 JTAG & Hardware functions

These commands are used to read, write and handle JTAG data and to set hardware states.

### 5.5.1 EMU\_CMD\_RESET\_TRST

#### Description

Resets TRST.

#### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0x02 |

Table 5.26: EMU\_CMD\_RESET\_TRST command overview

#### Response

-

#### Additional information

Activates TRST and releases it after 2ms.

#### Sample communication

Reset TRST command.

H->E: 02

## 5.5.2 EMU\_CMD\_SET\_SPEED

### Description

Sets the JTAG speed.

### Command

| Direction | Data    | Name  | Explanation   |
|-----------|---------|-------|---|
| H->E      | 1 * U8  | Cmd   | Command: 0x05   |
| H->E      | 1 * U16 | Speed | JTAG speed in kHz.<br>Example:<br>0x001E for 30kHz.<br>0xFFFF for adaptive clocking |

**Table 5.27: EMU\_CMD\_SET\_SPEED command overview**

### Response

-

### Sample communication

Set speed to adaptive clocking command.

H->E: 05 FF FF



## 5.5.3 EMU\_CMD\_SELECT\_IF

### Description

Select a target interface or retrieves the previously selected interface.

### Command

| Direction | Data   | Name   | Explanation   |
|-----------|--------|--------|---|
| H->E      | 1 * U8 | Cmd    | Command: 0xC7   |
| H->E      | 1 * U8 | SubCmd | 0xFF to return 32bit mask of available interfaces<br>0xFE to return previously selected interface number (0..31)<br>1..31 to select interface (0..31) |

Table 5.28: EMU\_CMD\_SELECT\_IF command overview

### Response

| Direction | Data    | Name     | Explanation   |
|-----------|---------|----------|---|
| H<-E      | 1 * U32 | Response | Response depends on SubCmd sent.<br>SubCmd == 0xFF: 32bit mask of available interfaces<br>SubCmd == 0xFE: Previously selected interface number (0..31)<br>SubCmd == 0..31: Previously selected interface number (0..31) |

Table 5.29: EMU\_CMD\_SELECT\_IF response overview

### Additional information

The interfaces that can be selected (if available) are described in the table below:

| Interface # | Define   |
|-------------|----------|
| 0           | TIF_JTAG |
| 1           | TIF_SWD  |

Table 5.30: EMU\_CMD\_SELECT\_IF interface overview

### Sample communication (requesting available interfaces)

Output command to check for available interfaces.

H->E: C7 FF

Returns available interfaces as 32bit mask (in this case JTAG and SWD are available).

H<-E: 03 00 00 00

### Sample communication (selecting SWD interface)

Output command to select SWD interface.

H->E: C7 02

Returns previously selected interface (in this case previously selected interface was JTAG).

H<-E: 01 00 00 00

## 5.5.4 EMU\_CMD\_SET\_KS\_POWER

### Description

Sets KS power on JTAG pin 19 to on or off temporarily.

### Command

| Direction | Data   | Name  | Explanation                                     |
|-----------|--------|-------|---|
| H->E      | 1 * U8 | Cmd   | Command: 0x08                                   |
| H->E      | 1 * U8 | OnOff | Set KS power on: 0x01<br>Set KS power off: 0x00 |

**Table 5.31: EMU\_CMD\_SET\_KS\_POWER command overview**

### Response

-

### Additional information

JTAG pin 19 may provide 5V power to a target system. For details, please refer to J-Link User manual.

### Sample communication

Set KS power to on command.

H->E: 08 01

## 5.5.5 EMU\_CMD\_HW\_CLOCK

### Description

Generates one clock and retrieves data from TDI.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xC8 |

**Table 5.32: EMU\_CMD\_HW\_CLOCK command overview**

### Response

| Direction | Data   | Name | Explanation    |
|-----------|--------|------|----------------|
| H<-E      | 1 * U8 | TDI  | Data from TDI. |

**Table 5.33: EMU\_CMD\_HW\_CLOCK response overview**

### Additional information

This command generates one clock and returns the value of TDI on falling edge of clock.

Default value of clock ping (TCK) is high. So this function generates a low pulse, followed by a high pulse on TCK.

### Sample communication

Generate one clock.

H->E: C8

Returns TDI data.

H<-E: 00

## 5.5.6 EMU\_CMD\_HW\_TMS0

### Description

Clears TMS signal.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xC9 |

Table 5.34: EMU\_CMD\_HW\_TMS0 command overview

### Response

-

### Sample communication

Clear TMS command.

H->E: C9

## 5.5.7 EMU\_CMD\_HW\_TMS1

### Description

Sets TMS signal.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xCA |

Table 5.35: EMU\_CMD\_HW\_TMS1 command overview

### Response

-

### Sample communication

Set TMS command.

H->E: CA

## 5.5.8 EMU\_CMD\_HW\_DATA0

### Description

Clears TDI signal.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xCB |

Table 5.36: EMU\_CMD\_HW\_DATA0 command overview

### Response

-

### Sample communication

Clear TDI command.

H->E: CB

## 5.5.9 EMU\_CMD\_HW\_DATA1

### Description

Sets TDI signal.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xCC |

**Table 5.37: EMU\_CMD\_HW\_DATA1 command overview**

### Response

-

### Sample communication

Set TDI command.

H->E: CC

## 5.5.10 EMU\_CMD\_HW\_JTAG

### Description

Handles the JTAG connection. It receives data for TDI and TMS and sends TDO data back.

### Command

| Direction | Data               | Name    | Explanation  |
|-----------|--------------------|---------|--|
| H->E      | 1 * U8             | Cmd     | Command: 0xCD  |
| H->E      | 1 * U16            | NumBits | Number of bits to transfer.  |
| H->E      | <NumBytes><br>* U8 | TMS     | Data for TMS. <i>NumBytes</i> calculates as follows:<br>$\text{NumBytes} = (\text{NumBits} + 7) \gg 3$ |
| H->E      | <NumBytes><br>* U8 | TDI     | Data for TDI. <i>NumBytes</i> calculates as follows:<br>$\text{NumBytes} = (\text{NumBits} + 7) \gg 3$ |

Table 5.38: EMU\_CMD\_HW\_JTAG command overview

### Response

| Direction | Data               | Name | Explanation  |
|-----------|--------------------|------|--|
| H<-E      | <NumBytes><br>* U8 | TDO  | TDO data. <i>NumBytes</i> calculates as follows:<br>$\text{NumBytes} = (\text{NumBits} + 7) \gg 3$ |

Table 5.39: EMU\_CMD\_HW\_JTAG response overview

### Additional information

This command is obsolete. Please use *EMU\_CMD\_HW\_JTAG3* on page 50.



## 5.5.11 EMU\_CMD\_HW\_JTAG2

### Description

Handles the JTAG connection. It receives data for TDI and TMS and sends TDO data back.

### Command

| Direction | Data               | Name    | Explanation  |
|-----------|--------------------|---------|--|
| H->E      | 1 * U8             | Cmd     | Command: 0xCE  |
| H->E      | 1 * U8             | DUMMY   | Dummy data in order to word align data transfers.  |
| H->E      | 1 * U16            | NumBits | Number of bits to transfer.  |
| H->E      | <NumBytes><br>* U8 | TMS     | Data for TMS. <i>NumBytes</i> calculates as follows:<br>$\text{NumBytes} = (\text{NumBits} + 7) \gg 3$ |
| H->E      | <NumBytes><br>* U8 | TDI     | Data for TDI. <i>NumBytes</i> calculates as follows:<br>$\text{NumBytes} = (\text{NumBits} + 7) \gg 3$ |

Table 5.40: EMU\_CMD\_HW\_JTAG2 command overview

### Response

| Direction | Data               | Name | Explanation  |
|-----------|--------------------|------|--|
| H<-E      | <NumBytes><br>* U8 | TDO  | TDO data. <i>NumBytes</i> calculates as follows:<br>$\text{NumBytes} = (\text{NumBits} + 7) \gg 3$ |

Table 5.41: EMU\_CMD\_HW\_JTAG2 response overview

### Additional information

This command is obsolete for J-Links with hardware version 5 and above. Please use *EMU\_CMD\_HW\_JTAG3* on page 50.

The hardware version can be retrieved using the command *EMU\_CMD\_GET\_HW\_VERSION* on page 34.

## 5.5.12 EMU\_CMD\_HW\_JTAG3

### Description

Handles the JTAG connection. If used with JTAG interface it receives data for TDI and TMS and sends TDO data back. If used with SWD interface it receives data for SWDIO (in) and DIR and sends SWDIO (out) data back.

### Command (JTAG interface)

| Direction | Data               | Name    | Explanation   |
|-----------|--------------------|---------|---|
| H->E      | 1 * U8             | Cmd     | Command: 0xCF   |
| H->E      | 1 * U8             | DUMMY   | Dummy data in order to word align data transfers.                                       |
| H->E      | 1 * U16            | NumBits | Number of bits to transfer.   |
| H->E      | <NumBytes><br>* U8 | TMS     | Data for TMS. <i>NumBytes</i> calculates as follows:<br>$NumBytes = (NumBits + 7) >> 3$ |
| H->E      | <NumBytes><br>* U8 | TDI     | Data for TDI. <i>NumBytes</i> calculates as follows:<br>$NumBytes = (NumBits + 7) >> 3$ |

Table 5.42: EMU\_CMD\_HW\_JTAG (JTAG interface) command overview

### Command (SWD interface)

| Direction | Data               | Name       | Explanation   |
|-----------|--------------------|------------|---|
| H->E      | 1 * U8             | Cmd        | Command: 0xCF   |
| H->E      | 1 * U8             | DUMMY      | Dummy data in order to word align data transfers.   |
| H->E      | 1 * U16            | NumBits    | Number of bits to transfer.   |
| H->E      | <NumBytes><br>* U8 | DIR        | Direction of data to transmit.<br><i>NumBytes</i> calculates as follows: $NumBytes = (NumBits + 7) >> 3$ .<br>0: In<br>1: Out |
| H->E      | <NumBytes><br>* U8 | SWDIO (in) | SWDIO input data. <i>NumBytes</i> calculates as follows: $NumBytes = (NumBits + 7) >> 3$                                      |

Table 5.43: EMU\_CMD\_HW\_JTAG (SWD interface) command overview

### Response (JTAG interface)

| Direction | Data               | Name   | Explanation   |
|-----------|--------------------|--------|---|
| H<-E      | <NumBytes><br>* U8 | TDO    | TDO data. <i>NumBytes</i> calculates as follows:<br>$NumBytes = (NumBits + 7) >> 3$ |
| H<-E      | 1 * U8             | Return | Return value:<br>0: O.K.<br>Everything else: Error occurred.                        |

Table 5.44: EMU\_CMD\_HW\_JTAG (JTAG interface) response overview

### Response (SWD interface)

| Direction | Data               | Name        | Explanation   |
|-----------|--------------------|-------------|---|
| H<-E      | <NumBytes><br>* U8 | SWDIO (out) | SWDIO output data. <i>NumBytes</i> calculates as follows: $NumBytes = (NumBits + 7) >> 3$ |
| H<-E      | 1 * U8             | Return      | Return value:<br>0: O.K.<br>Everything else: Error occurred.                              |

Table 5.45: EMU\_CMD\_HW\_JTAG (SWD interface) response overview

### Additional information

How the JTAG/SWD sequence is generated is beyond the scope of this manual.

This command is available for J-Links with hardware version 5 and above only.  
SWD support is available for J-Links with hardware version 6 and above only.

The hardware version can be retrieved using the command *EMU\_CMD\_GET\_HW\_VERSION* on page 34.

### **Sample communication**

Output command and JTAG TDI, TMS data to halt ARM7 CPU.

```
H->E: CF 00 48 00 86 E3 71 C0 01 00 00 00 0C 20 00 18 02 00 00 00 00 00
```

Returns TDO data.

```
H<-E: 20 00 08 00 48 00 00 00 00
```

Returns O.K.

```
H<-E: 00
```

## 5.5.13 EMU\_CMD\_HW\_JTAG\_WRITE

### Description

Handles the JTAG connection. If used with JTAG interface it receives data for TDI and TMS. No TDO data is sent back. If used with SWD interface it receives data for SWDIO (in) and DIR. No SWDIO (out) data is sent back. The status of the transmission is saved in a stick error flag. This flag can be requested via the `EMU_CMD_HW_JTAG_GET_RESULT` command. For more information about the `EMU_CMD_HW_JTAG_GET_RESULT` command, please refer to `EMU_CMD_HW_JTAG_GET_RESULT` on page 53

### Command (JTAG interface)

| Direction | Data               | Name    | Explanation  |
|-----------|--------------------|---------|--|
| H->E      | 1 * U8             | Cmd     | Command: 0xCF  |
| H->E      | 1 * U8             | DUMMY   | Dummy data in order to word align data transfers.  |
| H->E      | 1 * U16            | NumBits | Number of bits to transfer.  |
| H->E      | <NumBytes><br>* U8 | TMS     | Data for TMS. <i>NumBytes</i> calculates as follows:<br>$\text{NumBytes} = (\text{NumBits} + 7) \gg 3$ |
| H->E      | <NumBytes><br>* U8 | TDI     | Data for TDI. <i>NumBytes</i> calculates as follows:<br>$\text{NumBytes} = (\text{NumBits} + 7) \gg 3$ |

Table 5.46: EMU\_CMD\_HW\_JTAG\_WRITE (JTAG interface) command overview

### Command (SWD interface)

| Direction | Data               | Name       | Explanation  |
|-----------|--------------------|------------|--|
| H->E      | 1 * U8             | Cmd        | Command: 0xCF  |
| H->E      | 1 * U8             | DUMMY      | Dummy data in order to word align data transfers.  |
| H->E      | 1 * U16            | NumBits    | Number of bits to transfer.  |
| H->E      | <NumBytes><br>* U8 | DIR        | Direction of data to transmit.<br><i>NumBytes</i> calculates as follows: $\text{NumBytes} = (\text{NumBits} + 7) \gg 3$ .<br>0: In<br>1: Out |
| H->E      | <NumBytes><br>* U8 | SWDIO (in) | SWDIO input data. <i>NumBytes</i> calculates as follows: $\text{NumBytes} = (\text{NumBits} + 7) \gg 3$                                      |

Table 5.47: EMU\_CMD\_HW\_JTAG\_WRITE (SWD interface) command overview

### Additional information

How the JTAG/SWD sequence is generated is beyond the scope of this manual.

This command is available for J-Links with hardware version 6 and above only.

The hardware version can be retrieved using the command `EMU_CMD_GET_HW_VERSION` on page 34.

### Sample communication

Output command and JTAG TDI, TMS data to halt ARM7 CPU.

```
H->E: CF 00 48 00 86 E3 71 C0 01 00 00 00 0C 20 00 18 02 00 00 00 00 00
```

## 5.5.14 EMU\_CMD\_HW\_JTAG\_GET\_RESULT

### Description

Requests the sticky error bit from the emulator. This bit is set if a `EMU_CMD_HW_JTAG_WRITE` command failed. After the `EMU_CMD_HW_JTAG_GET_RESULT` command has been performed, the sticky error bit is cleared.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xD6 |

Table 5.48: EMU\_CMD\_HW\_JTAG\_GET\_RESULT command overview

### Response

| Direction | Data   | Name             | Explanation  |
|-----------|--------|------------------|--|
| E->H      | 1 * U8 | sticky error bit | Return value:<br>0: O.K.<br>Everything else: Error |

Table 5.49: EMU\_CMD\_HW\_JTAG\_GET\_RESULT command overview

### Sample communication

Output command to get the sticky error bit:

H->E: D6

Returns O.K.

E->H: 00

## 5.5.15 EMU\_CMD\_HW\_TRST0

### Description

Activates TRST signal.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xDE |

Table 5.50: EMU\_CMD\_HW\_TRST0 command overview

### Response

-

### Sample communication

Activate TRST command.

H->E: DE

## 5.5.16 EMU\_CMD\_HW\_TRST1

### Description

Deactivates TRST signal.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xDF |

**Table 5.51: EMU\_CMD\_HW\_TRST1 command overview**

### Response

-

### Sample communication

Deactivate TRST command.

H->E: DF

## 5.5.17 EMU\_CMD\_WRITE\_DCC

### Description

Writes to the CPU through JTAG using DCC.

### Command

| Direction | Data               | Name        | Explanation  |
|-----------|--------------------|-------------|--|
| H->E      | 1 * U8             | Cmd         | Command: 0xF1  |
| H->E      | 1 * U16            | NumBitsInit | TBD.   |
| H->E      | 1 * U16            | NumBitsStat | TBD.   |
| H->E      | 1 * U16            | NumBitsData | TBD.   |
| H->E      | 1 * U16            | BitPosStat  | TBD.   |
| H->E      | 1 * U16            | Timeout     | TBD.   |
| H->E      | 1 * U16            | NumItems    | TBD.   |
| H->E      | <Numbytes><br>* U8 | JTAG Data   | JTAG Data. <i>NumBytes</i> calculates as follows:<br>$NumBytes = 2 * ((NumBitsInit + 7) >> 3) + ((NumBitsStat + 7) >> 3) + ((NumBitsData + 7) >> 3)$ |
| H->E      | <NumItems><br>* U8 | Data        | Data content.  |

Table 5.52: EMU\_CMD\_WRITE\_DCC command overview

### Response

| Direction | Data   | Name   | Explanation   |
|-----------|--------|--------|---|
| H<-E      | 1 * U8 | Return | Return value of the command:<br>0x00: O.K.<br>0x01: Timeout |

Table 5.53: EMU\_CMD\_WRITE\_DCC response overview

### Additional information

Explanation of DCC data generation is beyond the scope of this document.



## 5.6 Target functions

These commands are used to act on a target hardware such as resetting it.

### 5.6.1 EMU\_CMD\_RESET\_TARGET

#### Description

Resets the target via hardware reset.

#### Communication

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0x03 |

**Table 5.54: EMU\_CMD\_RESET\_TARGET command overview**

#### Response

-

#### Additional information

Activates the target RESET line and releases it after 2ms.

#### Sample communication

Reset target command.

H->E: 03

## 5.6.2 EMU\_CMD\_HW\_RELEASE\_RESET\_STOP\_EX

### Description

Resets the CPU and halts as soon as possible. Means as soon as a criteria is matched.

### Command

| Direction | Data               | Name            | Explanation                                       |
|-----------|--------------------|-----------------|---|
| H->E      | 1 * U8             | Cmd             | Command: 0xD0                                     |
| H->E      | 1 * U16            | NumBytes        | Number of bytes to transfer.                      |
| H->E      | 1 * U16            | NumReps         | Number of repeats trying to halt the CPU.         |
| H->E      | 1 * U16            | CriteriaByteOff | Offset address where to check for criteria match. |
| H->E      | 1 * U16            | CriteriaMask    | Criteria mask.                                    |
| H->E      | 1 * U16            | CriteriaData    | Criteria to match.                                |
| H->E      | <NumBytes><br>* U8 | TMS             | TMS data.   |
| H->E      | <NumBytes><br>* U8 | TDI             | TDI data.   |

Table 5.55: EMU\_CMD\_HW\_RELEASE\_RESET\_STOP\_EX command overview

### Response

| Direction | Data               | Name | Explanation      |
|-----------|--------------------|------|------------------|
| H<-E      | <NumBytes><br>* U8 | TDO  | TDO return data. |

Table 5.56: EMU\_CMD\_HW\_RELEASE\_RESET\_STOP\_EX response overview

### Additional information

In order to halt an ARM7 or ARM9 CPU as soon as possible the emulator can decide if the CPU is halted by checking for a criteria. This eliminates the USB latency a decision by the host PC would cause. What happens is that RESET is pulled high and the emulator tries to halt the CPU with a halting sequence as soon as possible for [NumReps](#) tries. To check if halting the CPU was successful the emulator checks for a criteria. A pseudo code could look like the following code:

```
U16 i = 0;

ResetHigh();
do {
    i++;
    SendSequence();
} while(((Data & CriteriaMask) != CriteriaData) && (i <= NumReps))
```

This command is implemented for ARM7 and ARM9 cores only.

### 5.6.3 EMU\_CMD\_HW\_RELEASE\_RESET\_STOP\_TIMED

#### Description

Resets the CPU and halts as soon as its possible or the timeout expires.

#### Command

| Direction | Data               | Name            | Explanation                                       |
|-----------|--------------------|-----------------|---|
| H->E      | 1 * U8             | Cmd             | Command: 0xD1                                     |
| H->E      | 1 * U16            | NumBytes        | Number of bytes to transfer.                      |
| H->E      | 1 * U16            | TimeOut         | Timeout in ms.                                    |
| H->E      | 1 * U16            | CryteriaByteOff | Offset address where to check for criteria match. |
| H->E      | 1 * U16            | Criteria-Mask   | Criteria mask.                                    |
| H->E      | 1 * U16            | CriteriaData    | Criteria to match.                                |
| H->E      | <NumBytes><br>* U8 | TMS             | TMS data.   |
| H->E      | <NumBytes><br>* U8 | TDI             | TDI data.   |

Table 5.57: EMU\_CMD\_HW\_RELEASE\_RESET\_STOP\_TIMED command overview

#### Response

| Direction | Data               | Name | Explanation      |
|-----------|--------------------|------|------------------|
| H<-E      | <NumBytes><br>* U8 | TDO  | TDO return data. |

Table 5.58: EMU\_CMD\_HW\_RELEASE\_RESET\_STOP\_TIMED response overview

#### Additional information

For further information please refer to *EMU\_CMD\_HW\_RELEASE\_RESET\_STOP\_EX* on page 58. The only difference of this command is that the abort criteria is a timeout instead of a maximum repeat count.

## 5.6.4 EMU\_CMD\_HW\_RESET0

### Description

Activates target reset.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xDC |

Table 5.59: EMU\_CMD\_HW\_RESET0 command overview

### Response

-

### Sample communication

Activate target reset command.

H->E: DC

## 5.6.5 EMU\_CMD\_HW\_RESET1

### Description

Deactivates target reset.

### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xDD |

Table 5.60: EMU\_CMD\_HW\_RESET1 command overview

### Response

-

### Sample communication

Deactivate target reset command.

H->E: DD

## 5.6.6 EMU\_CMD\_GET\_CPU\_CAPS

### Description

Retrieves the capabilities of the target CPU.

### Command

| Direction | Data   | Name          | Explanation  |
|-----------|--------|---------------|--|
| H->E      | 1 * U8 | Cmd           | Command: 0xE9  |
| H->E      | 1 * U8 | Device-Family | Device family of the target CPU                                  |
| H->E      | 1 * U8 | Interface     | Target interface used to connect the emulator to the target CPU. |
| H->E      | 1 * U8 | Reserved      | Dummy  |
| H->E      | 1 * U8 | Reserved      | Dummy  |

Table 5.61: EMU\_CMD\_GET\_CPU\_CAPS command overview

### Response

| Direction | Data    | Name         | Explanation                                |
|-----------|---------|--------------|--|
| H<-E      | 1 * U32 | Capabilities | Functionality supported by the target CPU. |

Table 5.62: EMU\_CMD\_GET\_CPU\_CAPS response overview

### Additional information

The capability flags used are described in the table below:

| Bit | Define            | Explanation                          |
|-----|-------------------|--------------------------------------|
| 0   | CPU_CAP_RESERVED  | Always 1.                            |
| 1   | CPU_CAP_WRITE_MEM | Supports command "CPU_CMD_WRITE_MEM" |
| 2   | CPU_CAP_READ_MEM  | Supports command "CPU_CMD_READ_MEM"  |

Table 5.63: EMU\_CMD\_GET\_CPU\_CAPS flag overview

### Sample communication

Requesting the CPU capabilities.

```
H->E: E9 07 00 00 00
```

Returns no capabilities.

```
H<-E: 01 00 00 00
```

## 5.6.7 EMU\_CMD\_EXEC\_CPU\_CMD

### Description

Executes functions supported by the CPU.

### Command

| Direction | Data                         | Name               | Explanation  |
|-----------|------------------------------|--------------------|--|
| H->E      | 1 * U8                       | Cmd                | Command: 0xEA  |
| H->E      | 1 * U8                       | CPUCmd             | CPU command to be executed.  |
| H->E      | 1 * U8                       | SubCmd             | Sub command for the CPU command if any.  |
| H->E      | 1 * U8                       | Device-Family      | Device family of the target CPU  |
| H->E      | 1 * U8                       | Interface          | Target interface used to connect the emulator to the target CPU.                             |
| H->E      | 1 * U16                      | NumBytes           | Number of bytes to transmit for the configuration.   |
| H->E      | <NumBytes><br>* U8           | Config             | Configuration data.  |
| H->E      | 1 * U32                      | Addr               | Memory location to write to / read from.   |
| H->E      | 1 * U32                      | NumBytes-WriteRead | Number of bytes to write/read to/from given address.   |
| H->E      | <NumBytes-WriteRead><br>* U8 | Data               | Data to be written to the given memory address.<br>Only used with command CPU_CMD_WRITE_MEM. |

Table 5.64: EMU\_CMD\_EXEC\_CPU\_CMD command overview

### Response

| Direction | Data    | Name   | Explanation   |
|-----------|---------|--------|---|
| H<-E      | 1 * U32 | Result | Result of the write/read command.<br>0 : Everything O.K.<br>> 0 : Number of bytes that could not be read/written. |

Table 5.65: EMU\_CMD\_EXEC\_CPU\_CMD response overview

### Additional information

The list below gives an overview about the available CPU commands.

| Define            | Value | Explanation               |
|-------------------|-------|---------------------------|
| CPU_CMD_WRITE_MEM | 100   | Writes to target memory.  |
| CPU_CMD_READ_MEM  | 101   | Reads from target memory. |

Table 5.66: CPU command overview

## 5.6.8 EMU\_CMD\_WRITE\_MEM\_ARM79

### Description

Writes to target memory on ARM 7/9 targets.

### Command

| Direction | Data               | Name        | Explanation                             |
|-----------|--------------------|-------------|---|
| H->E      | 1 * U8             | Cmd         | Command: 0xF7                           |
| H->E      | 1 * U8             | TotalIRLen  | Total scanchain length.                 |
| H->E      | 1 * U8             | NumDevices  | Number of devices on scanchain.         |
| H->E      | 1 * U8             | DevicePos   | Devices before target device.           |
| H->E      | 1 * U8             | IRPre       | IR bits before target device.           |
| H->E      | 1 * U8             | IsARM9      | 0x00: ARM7<br>0x01: ARM9                |
| H->E      | 1 * U8             | IsBigEndian | 0x00: Little endian<br>0x01: Big endian |
| H->E      | 1 * U16            | Dummy       | Dummy field.                            |
| H->E      | 1 * U8             | SubCmd      | Typically 0x01 .                        |
| H->E      | 1 * U32            | Addr        | Target address in memory.               |
| H->E      | 1 * U32            | NumBytes    | Number of bytes to write.               |
| H->E      | <NumBytes><br>* U8 | Data        | Data that should be written to memory.  |

Table 5.67: EMU\_CMD\_WRITE\_MEM\_ARM79 command overview

### Response

| Direction | Data   | Name          | Explanation  |
|-----------|--------|---------------|--|
| H<-E      | 1 * U8 | Return status | 0x00: O.K.<br>0x01: Adaptive clocking timeout<br>0x02: Memory access timeout<br>0x03: Core error |

Table 5.68: EMU\_CMD\_WRITE\_MEM\_ARM79 response overview

### Sample communication

Writing 0xAB to address 0x12345678.

```
H->E: F7 04 01 00 00 00 00 00 00 01 78 56 34 12 01 00 00 00 AB
```

Returns O.K.

```
H<-E: 00
```



## 5.6.9 EMU\_CMD\_READ\_MEM\_ARM79

### Description

Reads from target memory on ARM 7/9 targets.

### Command

| Direction | Data    | Name        | Explanation                             |
|-----------|---------|-------------|---|
| H->E      | 1 * U8  | Cmd         | Command: 0xF8                           |
| H->E      | 1 * U8  | TotalIRLen  | Total scanchain length.                 |
| H->E      | 1 * U8  | NumDevices  | Number of devices on scanchain.         |
| H->E      | 1 * U8  | DevicePos   | Devices before target device.           |
| H->E      | 1 * U8  | IRPre       | IR bits before target device.           |
| H->E      | 1 * U8  | IsARM9      | 0x00: ARM7<br>0x01: ARM9                |
| H->E      | 1 * U8  | IsBigEndian | 0x00: Little endian<br>0x01: Big endian |
| H->E      | 1 * U16 | Dummy       | Dummy field.                            |
| H->E      | 1 * U8  | SubCmd      | Typically 0x01 .                        |
| H->E      | 1 * U32 | Addr        | Target address in memory.               |
| H->E      | 1 * U32 | NumBytes    | Number of bytes to read.                |

Table 5.69: EMU\_CMD\_READ\_MEM\_ARM79 command overview

### Response

| Direction | Data               | Name            | Explanation   |
|-----------|--------------------|-----------------|---|
| H<-E      | 1 * U8             | ReturnStatus    | 0x00: O.K.<br>0x01: Adaptive clocking timeout<br>0x02: Memory access timeout<br>0x03: Core error  |
| H<-E      | <NumBytes><br>* U8 | Data            | Data that is read from the memory.<br>This transaction only happens if <code>Return-Status == 0x00</code> or <code>NumBytes &lt;= 0x40</code> .                                 |
| H<-E      | 1 * U32            | ErrorAddr       | Address at which an error occurred.<br>This transaction only happens if <code>Return-Status != 0x00</code> or <code>NumBytes &gt; 0x40</code> .                                 |
| H<-E      | 1 * U32            | FailAccessWidth | Information about which data width could not be accessed in memory.<br>This transaction only happens if <code>Return-Status != 0x00</code> or <code>NumBytes &gt; 0x40</code> . |

Table 5.70: EMU\_CMD\_READ\_MEM\_ARM79 response overview

### Sample communication

Retrieve 1 byte from address 0x10 command.

H->E: F8 04 01 00 00 00 00 00 00 01 10 00 00 00 01 00 00 00

Returns O.K.

H<-E: 00

Returns data read from address 0x10.

H<-E: 18

## 5.7 Configuration functions

These commands are used to configure the emulator.

### 5.7.1 EMU\_CMD\_READ\_CONFIG

#### Description

Reads the emulator configuration from the J-Link.

#### Command

| Direction | Data   | Name | Explanation   |
|-----------|--------|------|---------------|
| H->E      | 1 * U8 | Cmd  | Command: 0xF2 |

**Table 5.71: EMU\_CMD\_READ\_CONFIG command overview**

#### Response

| Direction | Data     | Name                   | Explanation                                |
|-----------|----------|------------------------|--|
| H<-E      | 256 * U8 | Emulator configuration | Returns the actual emulator configuration. |

**Table 5.72: EMU\_CMD\_READ\_CONFIG response overview**

#### Sample communication

Retrieve emulator configuration command.

H->E: F2

Returns emulator configuration.

```
H<-E: 00 02 FF FF 01 00 00 00 FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

For further information please refer to *Emulator configuration* on page 16.

## 5.7.2 EMU\_CMD\_WRITE\_CONFIG

### Description

Writes the emulator configuration to the J-Link.

### Command

| Direction | Data     | Name                   | Explanation                     |
|-----------|----------|------------------------|---------------------------------|
| H->E      | 1 * U8   | Cmd                    | Command: 0xF3                   |
| H->E      | 256 * U8 | Emulator configuration | The new emulator configuration. |

Table 5.73: EMU\_CMD\_WRITE\_CONFIG communication overview

### Response

-

### Sample communication

Send emulator configuration command.

H->E: F3

Send emulator configuration.

```
H->E: 00 02 FF FF 01 00 00 00 FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

For further information please refer to *Emulator configuration* on page 16.



# Chapter 6

## Glossary

---

This chapter describes important terms used throughout this manual.

**Adaptive clocking**

A technique in which a clock signal is sent out by J-Link / J-Trace. J-Link / J-Trace waits for the returned clock before generating the next clock pulse. The technique allows the J-Link / J-Trace interface unit to adapt to differing signal drive capabilities and differing cable lengths.

**Application Program Interface**

A specification of a set of procedures, functions, data structures, and constants that are used to interface two or more software components together.

**Big-endian**

Memory organization where the least significant byte of a word is at a higher address than the most significant byte. See Little-endian.

**Device Under Test**

The device in the scan chain debugged at the moment.

**DUT**

See Device Under Test.

**EmbeddedICE**

The additional hardware provided by debuggable ARM processors to aid debugging.

**Halfword**

A 16-bit unit of information. Contents are taken as being an `unsigned integer` unless otherwise stated.

**Host**

A computer which provides data and other services to another computer. Especially, a computer providing debugging services to a target being debugged.

**ICache**

Instruction cache.

**ICE Extension Unit**

A hardware extension to the EmbeddedICE logic that provides more breakpoint units.

**ID**

Identifier.

**IEEE 1149.1**

The IEEE Standard which defines TAP. Commonly (but incorrectly) referred to as JTAG.

**In-Circuit Emulator (ICE)**

A device enabling access to and modification of the signals of a circuit while that circuit is operating.

**Instruction Register**

When referring to a TAP controller, a register that controls the operation of the TAP.

**IR**

See Instruction Register.

**Joint Test Action Group (JTAG)**

The name of the standards group which created the IEEE 1149.1 specification.

**Little-endian**

Memory organization where the least significant byte of a word is at a lower address than the most significant byte. See also Big-endian.

**Memory coherency**

A memory is coherent if the value read by a data read or instruction fetch is the value that was most recently written to that location. Obtaining memory coherency is difficult when there are multiple possible physical locations that are involved, such as a system that has main memory, a write buffer, and a cache.

**RESET**

Abbreviation of System Reset. The electronic signal which causes the target system other than the TAP controller to be reset. This signal is also known as "nSRST" "nSYSRST", "nRST", or "nRESET" in some other manuals. See also nTRST.

**nTRST**

Abbreviation of TAP Reset. The electronic signal that causes the target system TAP controller to be reset. This signal is known as nICERST in some other manuals. See also nSRST.

**Open collector**

A signal that may be actively driven LOW by one or more drivers, and is otherwise passively pulled HIGH. Also known as a "wired AND" signal.

**Processor Core**

The part of a microprocessor that reads instructions from memory and executes them, including the instruction fetch unit, arithmetic and logic unit, and the register bank. It excludes optional coprocessors, caches, and the memory management unit.

**RTCK**

Returned TCK. The signal which enables Adaptive Clocking.

**Scan Chain**

A group of one or more registers from one or more TAP controllers connected between TDI and TDO, through which test data is shifted.

**TAP Controller**

Logic on a device which allows access to some or all of that device for test purposes. The circuit functionality is defined in IEEE1149.1.

**Target**

The actual processor (real silicon or simulated) on which the application program is running.

**TCK**

The electronic clock signal which times data on the TAP data lines TMS, TDI, and TDO.

**TDI**

The electronic signal input to a TAP controller from the data source (upstream). Usually, this is seen connecting the J-Link / J-Trace Interface Unit to the first TAP controller.

**TDO**

The electronic signal output from a TAP controller to the data sink (downstream). Usually, this is seen connecting the last TAP controller to the J-Link / J-Trace Interface Unit.

**Test Access Port (TAP)**

The port used to access a device's TAP Controller. Comprises TCK, TMS, TDI, TDO, and nTRST (optional).

**Transistor-transistor logic (TTL)**

A type of logic design in which two bipolar transistors drive the logic output to one or zero. LSI and VLSI logic often used TTL with HIGH logic level approaching +5V and LOW approaching 0V.

**Watchpoint**

A location within the image that will be monitored and that will cause execution to stop when it changes.

**Word**

A 32-bit unit of information. Contents are taken as being an unsigned integer unless otherwise stated.



# Chapter 7

## Literature and references

---

This chapter lists documents, which we think may be useful to gain deeper understanding of technical details.

| Reference | Title  | Comments  |
|-----------|--|---|
| [UM08001] | SEGGER J-Link / J-Trace User's Guide.  | This document gives information about using the SEGGER J-Link / J-Trace ARM. It is publicly available from SEGGER ( <a href="http://www.segger.com">www.segger.com</a> ).                   |
| [UM08004] | SEGGER J-Link RDI User's Guide.  | This document contains some hardware specific informations which are thought of being helpful. It is publicly available from SEGGER ( <a href="http://www.segger.com">www.segger.com</a> ). |
| [IEEE]    | IEEE Standard Test Access Port and Boundary-Scan Architecture, IEEE SS94949. | This document contains the IEEE 1149.2000 JTAG standard. It is publicly available from IEEE ( <a href="http://www.ieee.org">www.ieee.org</a> ).   |

**Table 7.1: Literature and Refernces**

# Index

---

|                                      |    |   |    |
|--------------------------------------|----|---|----|
| <b>A</b>                             |    | <b>N</b>                                |    |
| Adaptive clocking .....              | 70 | nTRST .....                             | 71 |
| Application Program Interface .....  | 70 |   |    |
| <b>B</b>                             |    | <b>O</b>                                |    |
| Big-endian .....                     | 70 | Open collector .....                    | 71 |
| <b>D</b>                             |    | <b>P</b>                                |    |
| Device Under Test .....              | 70 | Processor Core .....                    | 71 |
| DUT .....                            | 70 |   |    |
| <b>E</b>                             |    | <b>R</b>                                |    |
| EmbeddedICE .....                    | 70 | RESET .....                             | 71 |
| EMU_CMD_HW_JTAG_GET_RESULT .....     | 53 | RTCK .....                              | 71 |
| EMU_CMD_HW_JTAG_WRITE .....          | 52 |   |    |
| <b>H</b>                             |    | <b>S</b>                                |    |
| Halfword .....                       | 70 | Scan Chain .....                        | 71 |
| Host .....                           | 70 | Support .....                           | 69 |
|                                      |    | Syntax, conventions used .....          | 5  |
| <b>I</b>                             |    | <b>T</b>                                |    |
| ICache .....                         | 70 | TAP Controller .....                    | 71 |
| ICE Extension Unit .....             | 70 | Target .....                            | 71 |
| ID .....                             | 70 | TCK .....                               | 71 |
| IEEE 1149.1 .....                    | 70 | TDI .....                               | 71 |
| In-Circuit Emulator .....            | 70 | TDO .....                               | 71 |
| Instruction Register .....           | 70 | Test Access Port (TAP) .....            | 72 |
| IR .....                             | 70 | Transistor-transistor logic (TTL) ..... | 72 |
| <b>J</b>                             |    | <b>W</b>                                |    |
| Joint Test Action Group (JTAG) ..... | 70 | Watchpoint .....                        | 72 |
|                                      |    | Word .....                              | 72 |
| <b>L</b>                             |    |   |    |
| Little-endian .....                  | 71 |   |    |
| <b>M</b>                             |    |   |    |
| Memory coherency .....               | 71 |   |    |

