Flasher ATE User Guide

Document: UM08035
Manual Version: 1.14
Revision: e
Date: March 30, 2020

A product of SEGGER Microcontroller GmbH

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 (SEGGER) assumes no responsibility for any errors or omissions. SEGGER makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. SEGGER specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

Copyright notice

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

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

Tel. +49-2173-99312-0
Fax. +49-2173-99312-28
E-mail: support@segger.com
Internet: www.segger.com

Manual versions

This manual describes the Flasher ATE device.

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

Print date: March 30, 2020

Manual version	Revision	Date	By	Description
1.14	e	200325	MF	updated target interface information
1.14	c	200320	MF	added target channel information for #Terminal and #RTT command.
1.12		190918	MF	corrected auto patch command in section “Commands and replies”
1.06	b	190419	MF	added pin out for terminal connection to target device
1.06	a	190405	MF	added STM8 adapter
1.05	a	190214	MF	update target power supply chapter, added hint to functional safety
1.05		190123	MF	added commands #rtton, #rttoff, #ipconfig
1.04	d	181112	MF	updated power supply chapter, corrected #auto patch command, added commands #auto nopatch, #fwversion, #fwversionmod
1.04	b	181031	MF	added firmware version command
1.04	a	181025	MF	added serial number command
1.03	a	180926	MF	updated command samples and Universal Flash Loader chapter
1.00	f	180724	MF	Corrected selmodule command parameters, relayouted some commands sections
1.00	e	180627	MF	Corrected poweron command parameters
1.00	4	180427	MF	Corrections after review
1.00	3	180403	MF	Corrections after review
1.00	2	180326	MF	Corrections after review
1.00	1	180323	MF	Extracted from Flasher Manual Extraction from Flasher manual, adding section setting up a project

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 C programming language.
- The target processor.
- DOS command line.

If you feel that your knowledge of C is not sufficient, we recommend *The C Programming Language* by Kernighan and Richie (ISBN 0--13--1103628), which describes the standard in C programming and, in newer editions, also covers the ANSI C standard.

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.
`Sample comment`	Comments 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 of contents

Chapter 1

Introduction

This chapter provides a short overview about the the Flasher ATE its features.

1.1 Flasher ATE overview

Flasher ATE is a programming tool for micro controllers with on-chip or external flash memory. Flasher ATE is designed for programming flash targets in stand-alone mode or remote controlled via a PC.

Flasher ATE connects to a PC using the USB / Ethernet / RS232 interface, running Windows 7, Windows 8 or Windows 10. In stand-alone mode, Flasher ATE can be driven by telnet interface, or via the RS232 interface (handshake control or ASCII interface). Flasher ATE has a 20-pin connector, which supports various interfaces. JTAG or SWD are supported natively. Other interfaces are available via adapters, sold separately.

1.1.1 Features of the Flasher ATE

- Supports up to 10 individual channels.
- Stand-alone programmer (Once set up, Flasher can be controlled without the use of a PC program).
- Flexible power supply (USB, DC).
- 128 MB memory for storage of target program on each module.
- Data files can be updated via the integrated FTP server.
- Various target interfaces

Supported cores	Supported target interfaces	Flash programming speed (depending on target hardware)
ARM7/ARM9/Cortex-M, AVR MEGA, AVR Tiny, AVR XMEGA, MSP430, Power PC e200z0, PIC16, PIC18, RX610, RX621, RX62N, RX62T, RX64, RH850, STM8	ICSP, I2C, JTAG, PDI, SPI, SWD, SWIM, UART	up to 300 Kbytes/ second

1.1.2 Working environment

General

The Flasher ATE can be operated from a PC with an appropriate software like J-Flash or in stand-alone mode.

Host System

IBM PC/AT or compatible CPU: 486 (or better) with at least 128MB of RAM, running Microsoft Windows 7, Windows 8, or Windows 10. It needs to have a USB, Ethernet, or RS232 interface available for communication with the Flasher ATE.

Power supply

The system is powered via the mainboard. The supply voltage needs to be in the range from 4.8V to 5.25V. If the supply voltage exceeds these limits, the mainboard immediately shuts down the power supply for the connected modules for safety reasons.

Current consumption (typical)

Mainboard	
5V DC connector (VMAIN)	100 mA (no flasher module connected, no USB, no Ethernet)
5V DC connector (VMAIN)	180 mA (no flasher module connected, USB and Ethernet connected)
USB (not applicable for hardware V1.1)	100 mA (no flasher module connected, no USB, no Ethernet)

USB (not applicable for hardware V1.1)	180 mA (no flasher module connected, USB and Ethernet connected)

Module	
5V via Flasher ATE Bus (VCC5V)	100 mA (no target power supply)
5V via Flasher ATE Bus (VCC5V)	220 mA (target power supply with 100mA)

> **Note**
>
> Many USB power supplies have a high drop on the supply voltage if they operate at their power limit. We recommend the DC connector and a DC power supply when operating the Flasher ATE.

Flasher PC-software (J-Flash)

The latest version of the J-Flash software, which is part of the J-Link software and documentation package, can be downloaded from our website:
https://www.segger.com/jlink-software.html For more information about using J-Flash, please refer to `UM08003_JFlash.pdf` (J-Flash user guide) which is also available for download on our website.

Universal Flash Loader Configurator software

The latest version of the Universal Flash Loader Configurator software can always be downloaded from our website:
https://www.segger.com/downloads/flasher

This software is only needed if your CPU requires the usage of the Universal Flash Loader.

Flasher ATE Getting Started

There is an additional Flasher ATE Getting Started (AN08007) available which can be downloaded here: https://www.segger.com/downloads/flasher.

1.2 Specifications

1.2.1 Specifications for Flasher ATE

General	
Supported OS	Microsoft Windows 7 Microsoft Windows 7 x64 Microsoft Windows 8 Microsoft Windows 8 x64 Microsoft Windows 10 Microsoft Windows 10 x64
Operating Temperature	+5 °C … +60 °C
Storage Temperature	-20 °C … +65 °C
Relative Humidity (non-condensing)	<90% rH
Safety notes	For indoor use only.
Mechanical mainboard	
Size (without cables)	108mm x 56mm x 20mm
Weight (without cables)	47g
Mechanical module	
Size (without cables)	108mm x 35mm x 20mm
Weight (without cables)	24g
Available interfaces	
USB Host interface	USB 2.0, high speed
Ethernet Host interface	10/100 MBit
RS232 Host interface	RS232 9-pin
Target connector (module)	JTAG 20-pin 0.1 pitch (interface adapters available)
JTAG Interface, Electrical	
Power Supply	USB powered or via external power supply (5V), max. 3A using 10 modules
Target interface voltage (VIF)	1.2 … 5V
Target supply voltage	3 - 15V (5V with no additional supply.)
Target supply current	100 mA (VCC5V - 5V) 400 mA (VTGT - 3 … 15V)
Reset Type	Open drain. Can be pulled low or tristated
Reset low level output voltage (VOL)	VOL ≤ 10% of VIF
For the whole target voltage range (1.2V ≤ VIF ≤ 5V)	
LOW level input voltage (VIL)	VIL ≤ 40% of VIF
HIGH level input voltage (VIH)	VIH ≥ 60% of VIF
For 1.2V ≤ VIF ≤ 3.6V	
LOW level output voltage (VOL) with a load of 10 kOhm	VOL ≤ 10% of VIF
HIGH level output voltage (VOH) with a load of 10 kOhm	VOH ≥ 90% of VIF
For 3.6 ≤ VIF ≤ 5V	
LOW level output voltage (VOL) with a load of 10 kOhm	VOL ≤ 20% of VIF

HIGH level output voltage (VOH) with a load of 10 kOhm	VOH ≥ 80% of VIF
JTAG Interface, Timing	
Max. JTAG speed	up to 15MHz
Data input rise time (Trdi)	Trdi ≤ 20ns
Data input fall time (Tfdi)	Tfdi ≤ 20ns
Data output rise time (Trdo)	Trdo ≤ 10ns
Data output fall time (Tfdo)	Tfdo ≤ 10ns
Clock rise time (Trc)	Trc ≤ 10ns
Clock fall time (Tfc)	Tfc ≤ 10ns

1.2.1.1 Supported CPU cores

The Flasher ATE supports the following CPU cores:

ARM Cortex

- Cortex-M0
- Cortex-M0+
- Cortex-M1
- Cortex-M23
- Cortex-M3
- Cortex-M33
- Cortex-M4
- Cortex-M7

ARM (legacy cores)

- ARM7
- ARM9
- ARM11

Microchip

- PIC16

Renesas RX

- RX610
- RX621
- RX62G
- RX62N
- RX62T

Freescale Power PC

- e200z0

ST

- STM8

TI

- MSP430

1.2.1.2 Supported Target interfaces

The Flasher ATE supports the following target interfaces:

- JTAG
- SWD
- SWIM

Chapter 2

Working with the Flasher ATE

This chapter describes functionality and how to use the Flasher ATE.

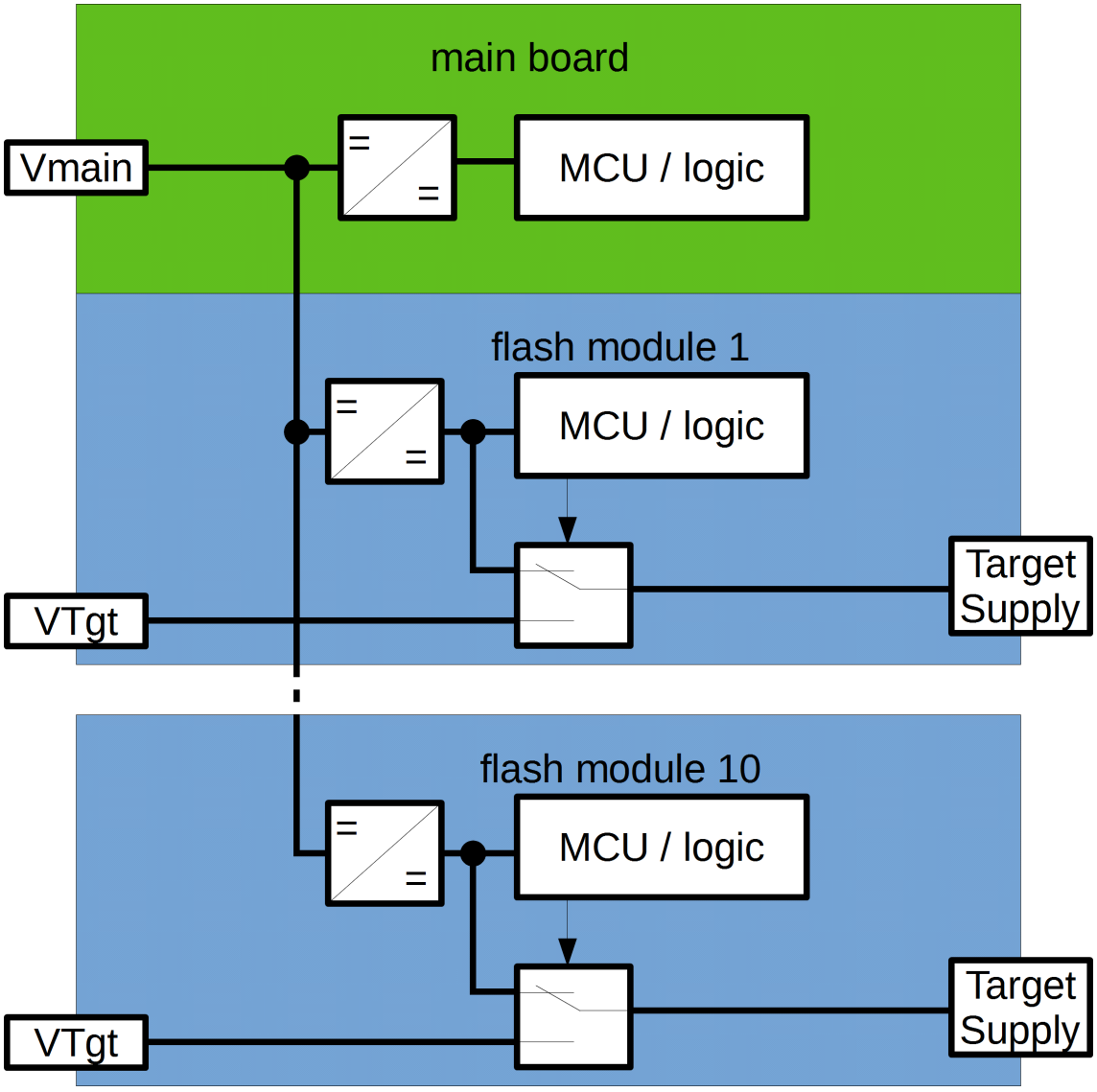
2.1 The Flasher ATE

The Flasher ATE has been designed to be used in conjunction with automated test equipment (ATE). It is modular and scalable from 1 to 10 individual programming modules.

2.1.1 Power supply

2.1.1.1 Mainboard and module power supply

The Flasher ATE mainboard and the modules are powered either by a dedicated power connector (VMAIN) or by USB. The power source may not be switched while the Flasher ATE is running.



2.1.1.2 Target power supply

For target power supply, the Flasher ATE offers two options. Additionally, a power discharge option is available. The target is electrically isolated from the mainboard’s power supply.

The first option is to use the VTGT connector. It is possible to provide any voltage between 3 and 15V, however the target interface is limited to 1.2V - 5V signal voltage. The VTGT connector is directly coupled to the target, so any electrical isolation has to be done in the external power supply.

The second option is to use the module’s internal power source VCC5V, which is able to deliver up to 100mA to the target.

The power discharge option applies, when the operation has finished. A 39 ohms resistor is used to discharge any remaining charge on the target’s capacitors. As the thermal load of the resistor is limited, the target’s capacity may not exceed 27mF at 15V or 250mF at 5V supply voltage.

> **Note**
>
> Flasher ATE programming modules provide functional isolation among each other. **Do not use with hazardous voltages to avoid risk of electrical shock and fire.**

2.2 Setting up the IP interface

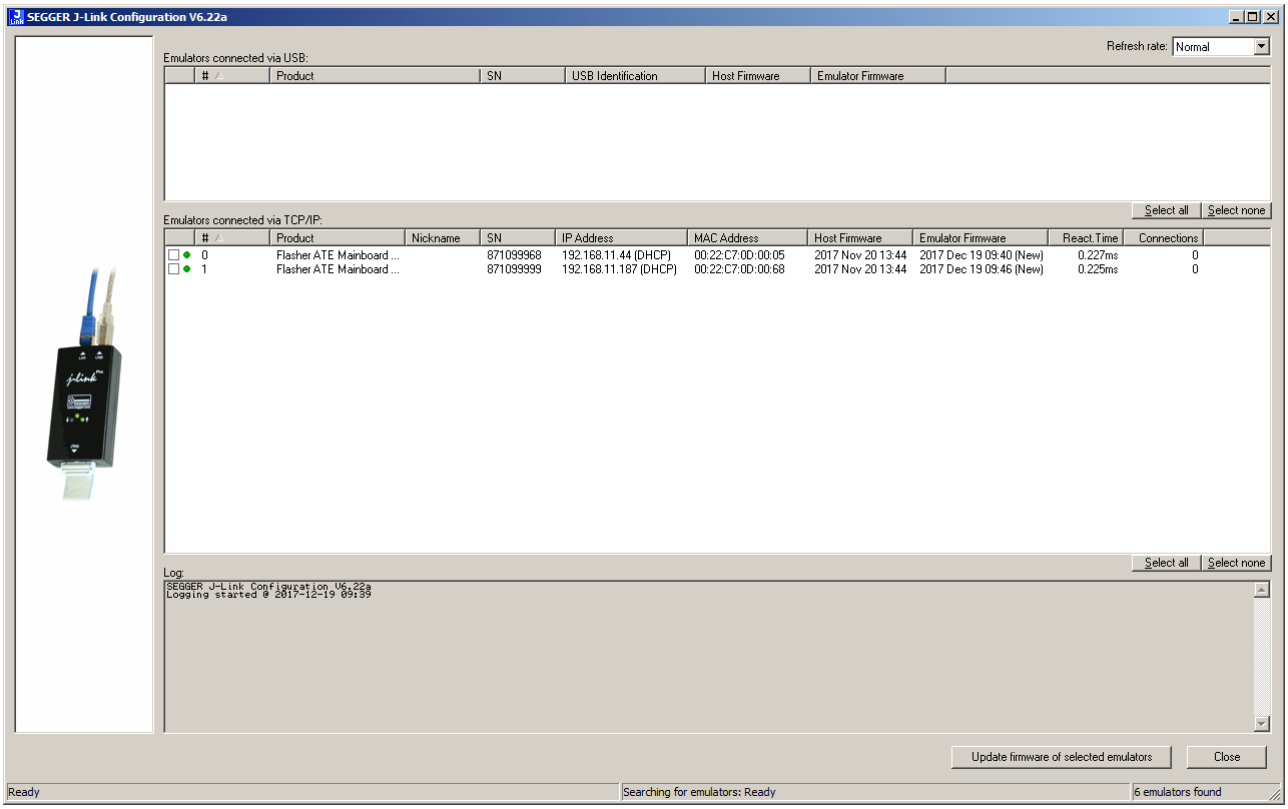
The Flasher ATE comes with an additional Ethernet interface to communicate with the host system. The Flasher ATE has a built-in webserver which allows some basic setup of the emulator, e.g. configuring a default gateway which allows using it even in large intranets.

2.2.1 Connecting for the first time

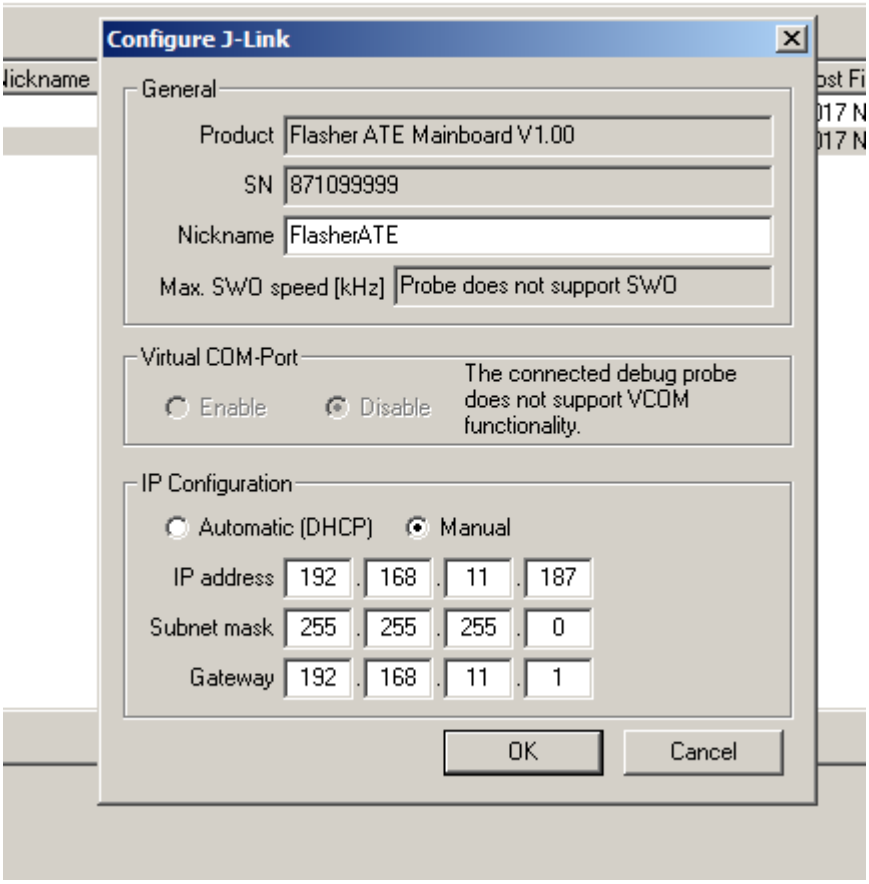
When connecting Flasher ATE for the first time, it attempts to acquire an IP address via DHCP. The recommended way for finding out which IP address has been assigned to Flasher ATE is to use the J-Link Configurator. The J-Link Configurator is a small GUI-based utility which shows a list of all emulators that are connected to the host PC via USB and Ethernet.

For more information about the J-Link Configurator, please refer to `UM08001_JLink.pdf` (J-Link / J-Trace user guide), chapter *Setup*, section *J-Link Configurator*.

The picture shows the J-Link Configurator with the detected Flasher ATEs. Select the one you want to configure.



You may configure your Flasher ATE to use your preferred IP setting. This is done by selecting the Flasher ATE in the list, opening the context menu with a right click on the list entry, and choosing **Configure** from the context menu. Enter your required IP settings in the dialog box that opens up, e.g. as shown in the next picture. Confirm the IP settings by pressing OK.



Note

If your computer is connected to a wireless LAN, some routers do not forward the broadcast messages used to find the Flasher ATE from LAN to WLAN. In that case either use a LAN connection for your computer or connect the Flasher ATE via USB to your computer and choose the Flasher ATE in the list of connected USB devices.

2.3 Operating modes

The Flasher ATE is able to operate in the following modes:

- remote controlled mode
- handshake mode

Definition of remote-controlled mode

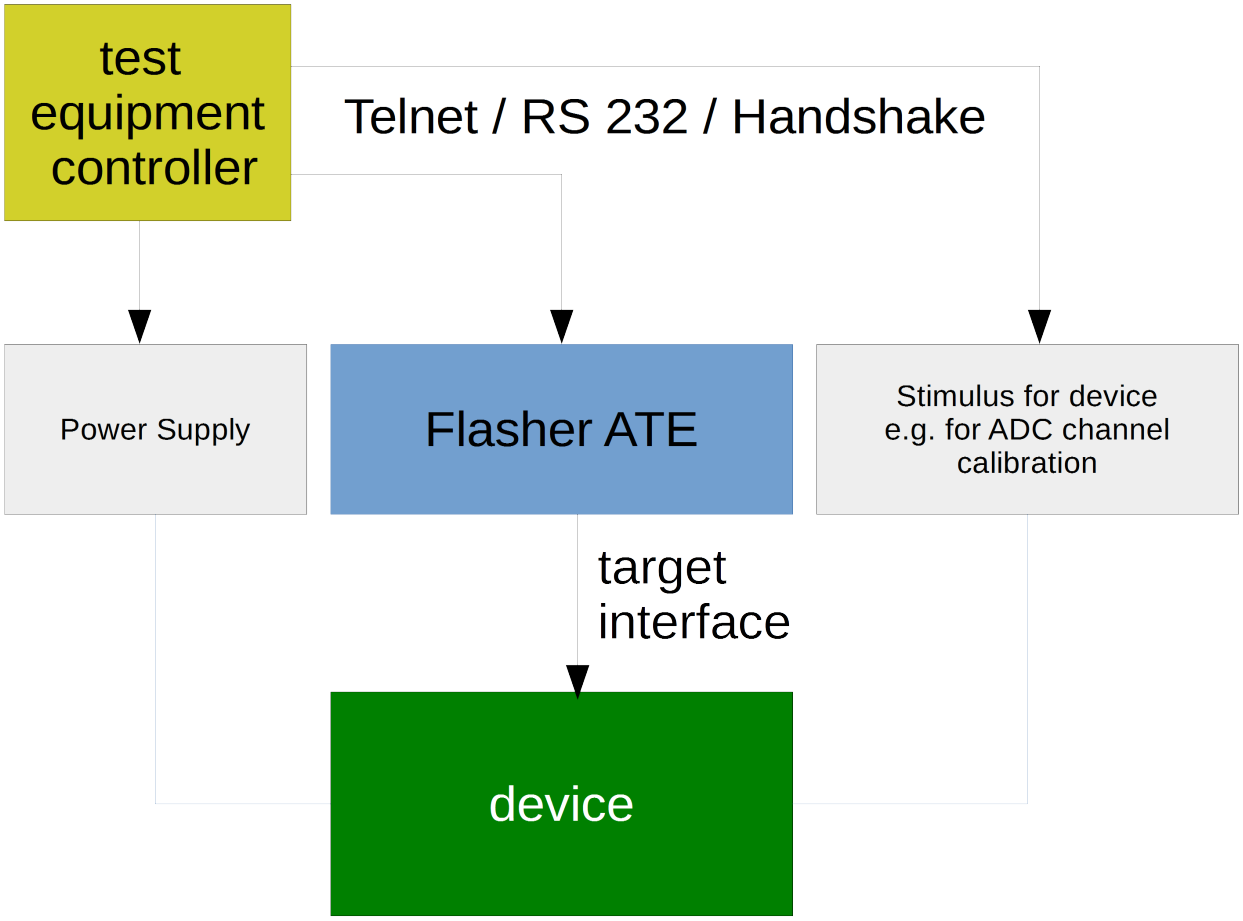
The remote-controlled mode is the recommended mode of operation. The Flasher ATE will receive its commands via an interface and will report the results to the caller. That is how the caller can check if the programming was done successfully or not.

Definition of handshake mode

The stand-alone mode is controlled the 3 handshake lines.

2.3.1 Remote-controlled mode

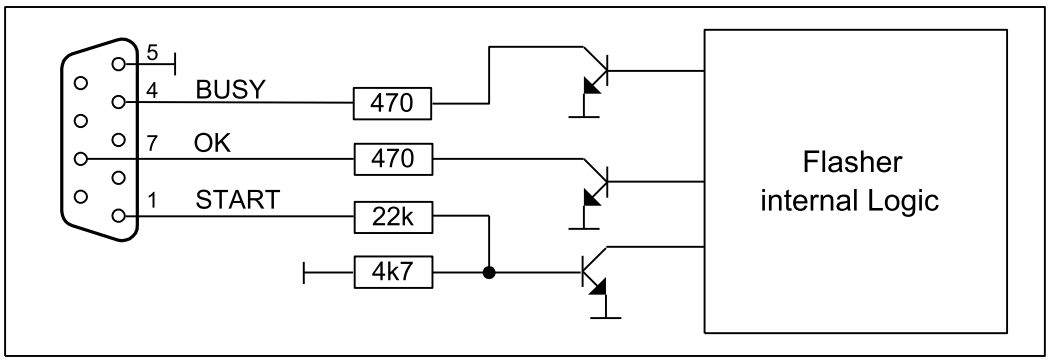
The remote-controlled mode provides a way to integrate the Flasher ATE into a production environment and control it via a communication protocol. Therefore two physical interfaces can be used: ethernet or RS232. In this setup, the Flasher ATE provides detailed status information which can be used to verify the success of the programming sequence, optimize the production setup, and identify errors. The following picture shows the Flasher ATE integration into a production environment.

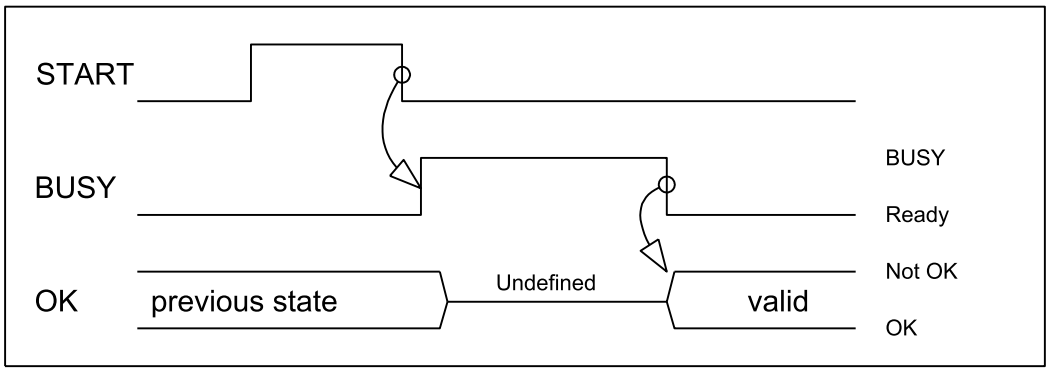


In order to use Flasher in “remote-controlled mode”, it has to be configured first, as described in *Setting up Flasher for remote-controlled mode* on page 31.

2.3.2 Handshake mode

The handshake mode provides a simple way to start the flashing of the connected targets. The pins 1, 4, 5 and 7 of the RS232 connector can be used to e.g. connect a start button and two indicator LEDs for programming control. The following picture show the Flasher ATE internal logic and the signal states during the programming sequence.





Note

The Flasher ATE will not provide any detail error codes during this mode. We recommend to use the remote controlled mode for mass production.

In order to use Flasher in “handshake mode”, it has to be configured first, as described in *Setting up Flasher for handshake mode* on page 33.

2.4 LED status indicators

The Flasher ATE uses different LEDs, see the following tables.

2.4.1 Mainboard LED indicators

#	Status of mainboard LEDs	Meaning
USB	GREEN high frequency blinking (~ 10Hz)	The mainboard is waiting for USB enumeration. As soon as USB has been enumerated, the green LED stops flashing and is switched to constant green.
1	USB GREEN constant	The mainboard has completed USB enumeration or no USB connection present.
2	USB RED constant	The mainboard is in bootloader mode.
3	READY/BUSY GREEN	No operation, the Flasher is ready.
4	READY/BUSY RED	Flashing operation in progress on at least one module.
5	PASS/ERROR GREEN	The last operation was successful on all modules.
6	PASS/ERROR RED	The last operation has failed on at least one module.
7	VMAIN GREEN	VMAIN is used as module power supply.
8	VUSB GREEN	VUSB is used as module power supply[1].

[1] The VUSB LED does not work on mainboards hardware version 1.0. This has no influence on functionality.

2.4.2 Module LED indicators

LED	Status	Meaning
STATUS	GREEN short flashing	The module waits for a start trigger.
1	STATUS GREEN slow blinking	Flashing operation in progress: • Erasing (blinking at 6.25 Hz) • Programming (blinking at 1.67 Hz) • Verifying (blinking at 5 Hz)
2	STATUS RED constant	a) The module is in bootloader mode. b) The last operation has failed.
3	5V GREEN	Internal power supply for 5V is working.
4	3.3V GREEN	Internal power supply for 3.3V is working.
5	VCC5V GREEN	Target power is enabled via internal supply.
6	VTGT GREEN	Target power is enabled via external supply.

2.5 Flasher ATE storage

The Flasher ATE stores the configuration and programming data on its internal memory. This can be accessed with an FTP-client. Each module has capacity of approximately 126MB for data and configuration files.

The files are stored on the Flash Modules. They are mounted via the mainboard in sub-folders named “Module.xxx”, with xxx being the number in the flash module chain, e.g. “Module.001” for the first module. The first module next to the mainboard is assigned the number one and the last module is assigned the number ten.

2.6 UART to TCP transceiver

The Flasher ATE includes an UART to TCP transceiver. Each module is accessible via a connection to a module-specific TCP port.

Module	TCP port
#1	41
#2	42
#3	43
#4	44
#5	45
#6	46
#7	47
#8	48
#9	49
#10	50

While the transceiver is switched off, status messages from the module are sent out to the terminal. If the transceiver is active, incoming data on pin 17 on the debug interface is sent to the TCP connection, and incoming data on the TCP connection is sent to pin 5 on the debug interface. The parameters for the UART configuration are handed over when activating the transceiver mode with the command *#TERMINAL <module(s)> <Baudrate>,<Databits>,<Parity>,<Stopbits>* on page 67.

> **Note**
>
> The transceiver cannot be enabled while the module is programming. Also programming, erasing etc. can not be started if the transceiver mode is active.

The UART to TCP transceiver supports:

- baud rate: 2,400 up to 115,200,
- parity: none, even and odd,
- full duplex mode.

2.7 Log files

The Flasher ATE writes log files. Each flash module writes the success or error into the log file.

The log files are stored in the module folder of the corresponding flash module. They can be downloaded using the FTP service.

2.8 Newline encoding

In general, for all patch files, init files etc. Flasher ATE supports both newline encodings:

- Windows: \r\n
- Unix/Mac: \n

All parser functionality etc. are written to be independent from the host operating system.

Chapter 3

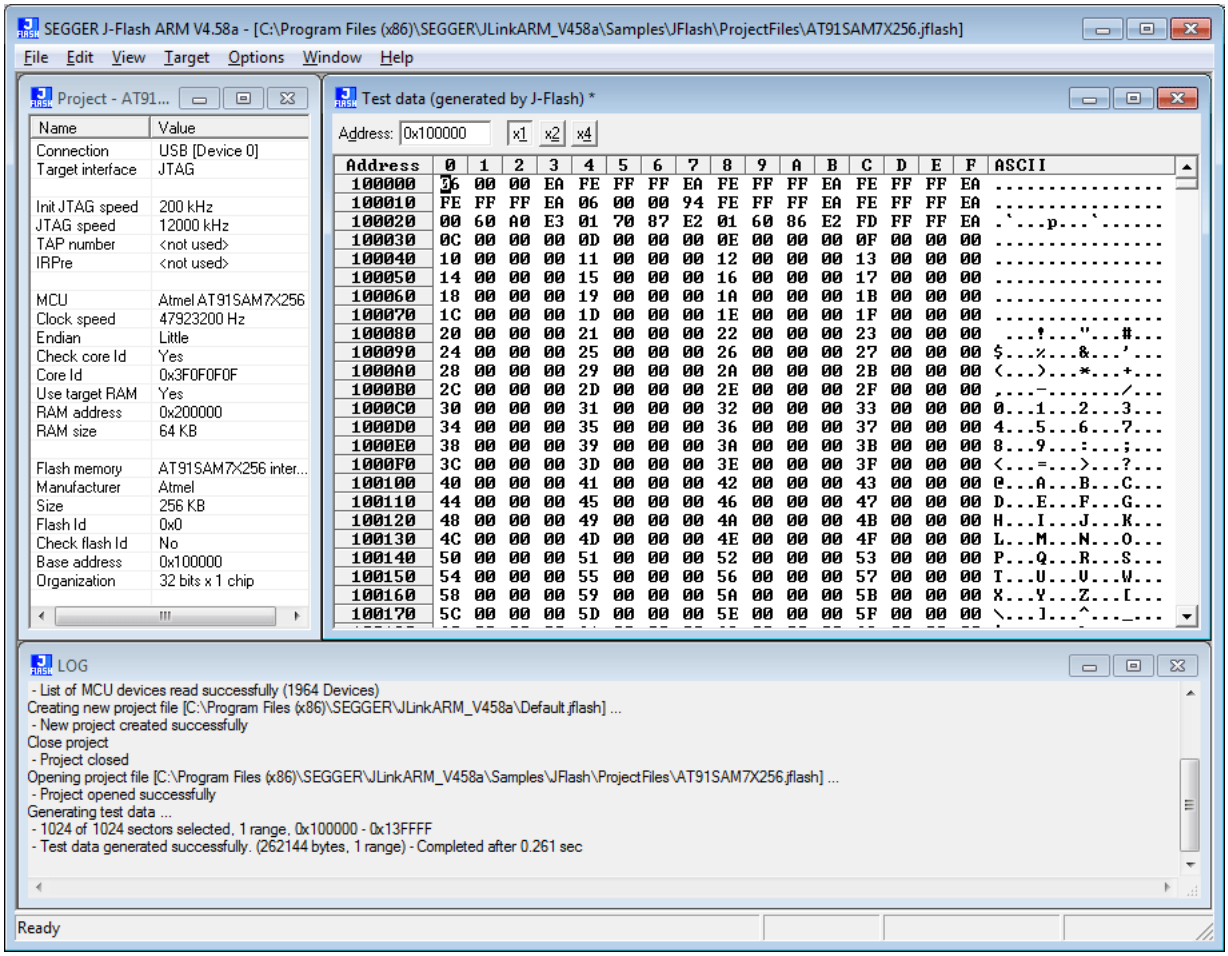
Setting up a project for the Flasher ATE

3.1 Setting up Flasher ATE for remote-controlled mode

In order to set up the Flasher ATE for the remote-controlled mode it needs to be configured once using the J-Flash software. For more information about J-Flash, please refer to the *J-Flash User Guide*.

After starting J-Flash, open the appropriate J-Flash project for the target the Flasher ATE shall be configured for, by selecting **File** -> **Open Project**. If J-Flash does not come with an appropriate sample project for the desired hardware, a new project needs to be created by selecting File -> **File** -> **New Project**.

After the appropriate project has been opened / created, the data file which shall be programmed needs to be loaded, by selecting **File** -> **Open**. After this J-Flash should look like in the screenshot below.



Before downloading the configuration (project) and program data (data file) to the Flasher ATE, both files need to saved to disk by **File** -> **Save Flasher Config File** and **File** -> **Save Flasher Data File**.

Next, upload the configuration file and the data file with an FTP client into the Flasher ATE module folder(s). If the J-Flash tool also generates a .pex file this also needs to be loaded into the module folder(s). But it must be located in a subfolder which must be named like your project. E.g. if your project is named MyTest with the project files MyTest.cfg and MyTest.dat, then the subfolder must be named MyTest.

From now on, the Flasher ATE can be used in stand-alone mode (without host PC interaction) for stand-alone programming.

3.2 Setting up Flasher ATE for handshake mode

In order to set up the Flasher ATE for the handshake mode it needs to be configured once using the J-Flash software. For more information about J-Flash, please refer to the *J-Flash User Guide*.

Please follow the steps in chapter *Setting up Flasher for remote-controlled mode* on page 31 first.

In addition you have to tell the Flasher ATE which project shall be programmed. Therefore you can directly edit the FLASHER.ini file in each modules folder, or connect to the Flasher ATE and use the "#select" command.

The `FLASHER.ini` file might look like as follows if your project is named MyTest:

```
[FILES]
DataFile = "MyTest.DAT"
ConfigFile = "MyTest.CFG"
```

From now on, the Flasher ATE can be used in stand-alone mode (without host PC interaction) for stand-alone programming.

3.3 Universal Flash Loader mode

As an alternative to the stand-alone mode, configured via J-Flash, there is the Universal Flash Loader mode. While the normal stand-alone mode relies on using the debug interface of the device, the Universal Flash Loader mode uses device or vendor specific programming interfaces and protocols and therefore it is independent of the CPU core.

3.3.1 Preparing manually

The Universal Flash Loader uses an initialization file (*.UNI), a device specific flash programming algorithm (*.PEX) and a data file (*.HEX, *.MOT, *.BIN or *.DAT).

3.3.1.1 Configuration

The initialization file basically is split into three parts. The first part, the section [DEVICE], controls the generic behavior of the Universal Flash Loader. It specifies which protocol driver and data file to use. It allows enabling and configuring target power and it defines which actions to perform. The second part consists of one or more [BANKx] sections, which contain information about the memories. The third part, the section [CONFIG], includes configuration settings for the protocol driver.

An .ini file might look as follows:

```
[OPTIONS]
TargetPower = "0"
ChipErase   = "0"

[TASKS]
CheckBlank  = "1"
Erase       = "1"
Program     = "1"
Verify      = "1"
Secure      = "1"

[DEVICE]
Algo        = "RL78.PEX"
Data        = "ALL_6k.mot"
Offset      = "0x00000000"

[BANK0]
; Code Flash
Base = "0x00000000"
Size = "0x00004000"
Sect = "0x00000400"

[BANK1]
; Data Flash
Base = "0x000F1000"
Size = "0x00000800"
Sect = "0x00000400"

[CONFIG]
BaudRate = "1000000"
ClearConfigOnConnect = "0x00"
Security = "0xFF"
ShieldStart = "0x0000"
ShieldEnd = "0x000F"
```

[OPTIONS]

TargetPower

If set to a value >0, power is applied to the target. The value defines the delay (in ms) after enabling the target power supply and before starting to communicate with the target.

ChipErase

If set to 1, the chip erase function is called for erasing the chip.

> **Note**
>
> Do not enable this setting if the flash programming algorithm does not support chip erase.

[TASKS]

CheckBlank

Defines if a blank check should be performed before erasing a sector.

Erase

Defines if the sector should be erased before programming.

Program

Defines if the sector should be programmed.

Verify

Defines if the sector should be verified after programming.

Secure

Defines if the device should be secured or protected against read-out after verifying.

[DEVICE]

Algo

File name of the flash programming algorithm. This file is provided by SEGGER and will typically support a series of devices.

Data

File name of the data file to program. The flasher supports the Flasher DTA, the Intel HEX, the Motorola S-Record and the binary file format. Flasher DAT files are generated by J-Flash and offer high performance together with high flexibility. The other file formats produce a small overhead, because they have to be parsed before the data can be programmed.

Offset

Offset to apply when programming a binary data file. Unless specified differently, binary files start at offset `0x00000000`.

[BANKx]

x blocks with configuration data for the flash banks. All three parameters (Base, Size and Sect) are mandatory.

Base

Base address of the flash bank.

Size

Total size of the flash bank.

Sect

Sector size of the flash bank.

[CONFIG]

This section includes specific configuration data for the flash programming algorithm. There are no general parameters.

> **Note**
>
> The data file must be organized in ascending address order. Gaps can be included. But descending addresses will result in programming errors. You can sort the data files by loading them into the J-Flash tool and saving it as a new file.

3.3.1.2 Configuration Data for Microchip AVR

The Microchip AVR devices do not require any configuration data.

Pinout

Pins are connected as follows:

Flasher Interface	Signal
Pin 1	VTref
Pin 2	VTref
Pin 4	GND
Pin 6	GND
Pin 8	GND
Pin 9	TOOL0
Pin 10	GND
Pin 12	GND
Pin 14	GND
Pin 15	nRESET
Pin 16	GND
Pin 18	GND
Pin 20	GND

3.3.1.3 Configuration Data for Microchip PIC

The Microchip PIC devices do not require any configuration data.

Pinout

Pins are connected as follows:

Flasher Interface	Signal
Pin 1	VTref
Pin 2	VTref
Pin 4	GND
Pin 6	GND
Pin 8	GND
Pin 9	TOOL0
Pin 10	GND
Pin 12	GND
Pin 14	GND
Pin 15	nRESET
Pin 16	GND
Pin 18	GND
Pin 20	GND

3.3.1.4 Configuration Data for Renesas RL78/G10

The RL78/G10 devices do not require any configuration data.

3.3.1.5 Configuration Data for Renesas RL78 (except RL78/G10)

BaudRate

The baud rate used for programming. Possible values are 115,200, 250,000, 500,000 and 1,000,000.

ClearConfigOnConnect

If this is set to 1, the first sector holding the configuration is cleared on connect. This will especially reset the clock configuration to its default value allowing a higher programming speed.

Security

Security configuration byte:

Item	Contents
Bit 7	Fixed to 1
Bit 6	Fixed to 1
Bit 5	Fixed to 1
Bit 4	Programming disable flag (1: Enable programming, 0: Disable programming)
Bit 3	Fixed to 1
Bit 2	Block erase disable flag (1: Enable block erase, 0: Disable block erase)
Bit 1	Boot block cluster rewrite disable flag (1: Enable boot block cluster rewrite, 0: Disable boot block cluster rewrite)
Bit 0	Fixed to 1

> **Note**
>
> Any bit set to 0 cannot be set to 1 again.

ShieldStart

Flash shield window start block number

ShieldEnd

Flash shield window end block number

Pinout

Pins are connected as follows:

Flasher Interface	Signal
Pin 1	VTref
Pin 2	VTref
Pin 4	GND
Pin 6	GND

Flasher Interface	Signal
Pin 8	GND
Pin 9	TOOL0
Pin 10	GND
Pin 12	GND
Pin 14	GND
Pin 15	nRESET
Pin 16	GND
Pin 18	GND
Pin 20	GND

3.3.1.6 Configuration Data for ST STM8

HighSpeed

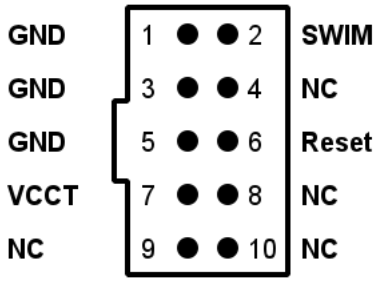
The interface mode (0 = low speed / 1 = high speed) used for communication.

ROP

Read out protection configuration byte: Depending on the target, `0x00` or `0xAA`.

SectorSize

Sector size: Depending on the target: “64” for low density flash memory, “128” for medium or high density flash memory.



Pins are connected as follows:

Flasher Interface	Signal
Pin 1	VCCT
Pin 2	VCCT
Pin 4	GND
Pin 6	GND
Pin 8	GND
Pin 9	SWIM
Pin 10	GND
Pin 12	GND
Pin 14	GND
Pin 15	Reset
Pin 16	GND
Pin 18	GND
Pin 20	GND

Note

As the STM8’s option bytes are part of the data image, the data image must not enable the read out protection for the device in order to allow verification after programming. The read out protection can be set finally by enabling the step “Secure”. This function only changes the ROP option byte to the appropriate value.

3.3.1.7 Configuration Data for TI MSP430: 1xx, 2xx and 4xx series

JTAGSpeed

The JTAG interface speed used for communication.

ClocksMassErase

Number of clocks required for a mass erase. This value depends on the device, please refer to the data sheet.

ClocksSegmentErase

Number of clocks required for a segment erase. This value depends on the device, please refer to the data sheet.

ClocksProgram

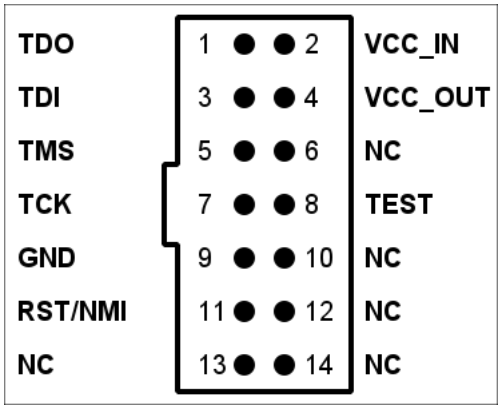
Number of clocks required for a word programming operation. This value depends on the device, please refer to the data sheet.

3.3.1.8 Configuration Data for TI MSP430: 5xx and 6xx series

JTAGSpeed

The JTAG interface speed used for communication.

Pinout



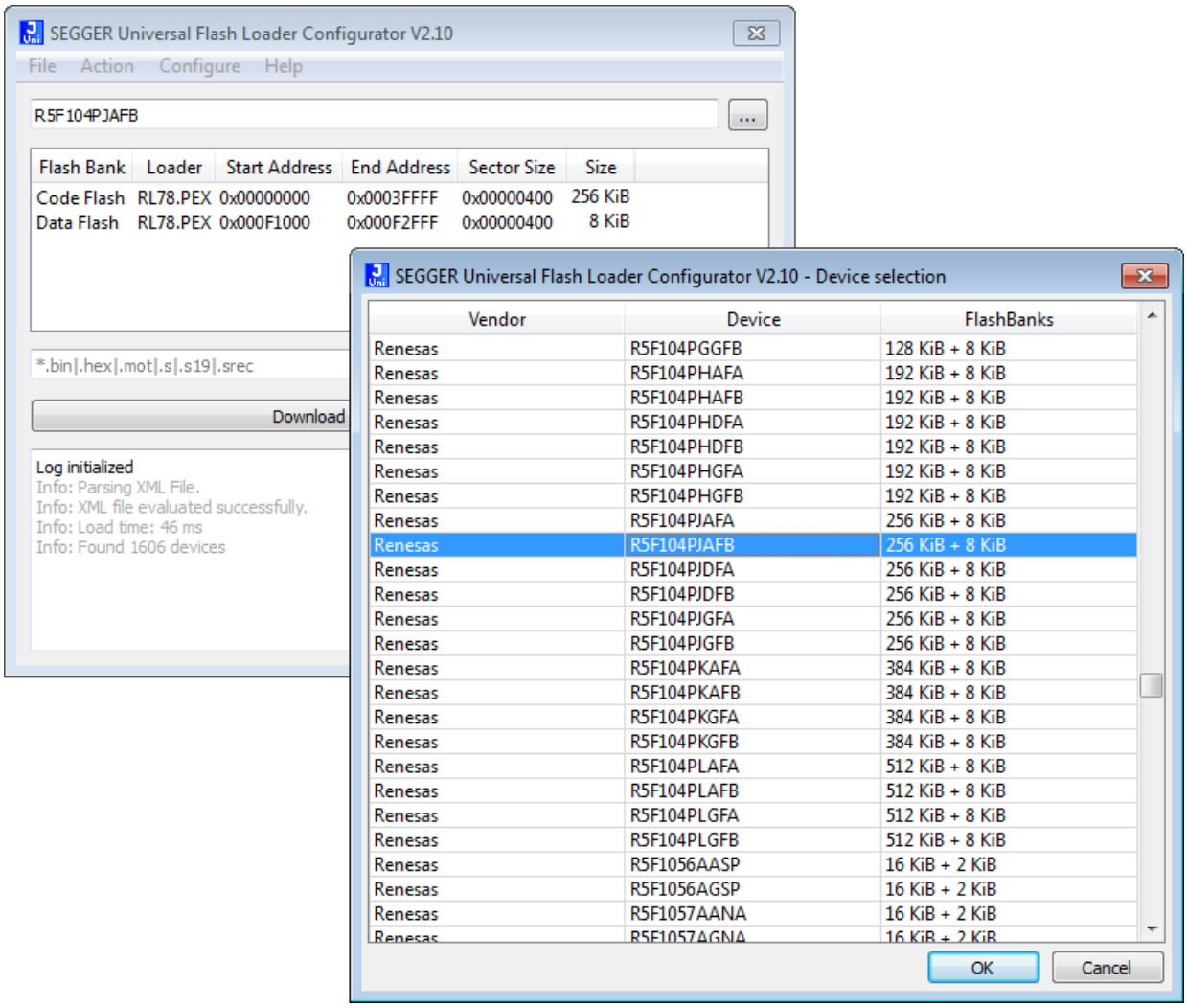
Pins are connected as follows:

Flasher Interface	Signal
Pin 1	VCC_OUT
Pin 3	TEST
Pin 4	GND
Pin 5	TDI
Pin 6	GND
Pin 7	TMS
Pin 8	GND
Pin 9	TCK
Pin 10	GND
Pin 12	GND
Pin 13	TDO
Pin 14	GND
Pin 15	RST/NMI

Flasher Interface	Signal
Pin 16	GND
Pin 18	GND
Pin 20	GND

3.3.2 Preparing using the PC utility

In order to set up Flasher for the Universal Flash Loader mode, a PC utility called SEGGER Universal Flash Loader Configurator is available for download.



The Universal Flash Loader Configurator comes with a large list of devices and flash programming algorithms. If you are going to use a device from one of the supported families which currently is not available in the utility, feel free to contact the support.

The functionality for downloading configuration and data files to the Flasher PRO is built-in. For the Flasher ATE, the files need to be manually exported using “Save Flasher UNI file…” and then transferred to the appropriate modules via FTP.

3.3.3 Connection for Device with no special Adapter

3.3.3.1 Connecting a I2C Device

Flasher ATE pins need to be connected as follows:

Flasher Interface	Flasher Signal Name	I2C Device Signal Name
Pin 1	VTRef	VCC
Pin 7	TMS/SWDIO	SDA
Pin 9	TCK/SWCLK	SCL
Pin 4,6,8,10,12,14,16,18 or 20	GND	GND

Chapter 4

Serial number handling

This chapter describes how the Flasher ATE deals with serial numbers.

4.1 Serial number programming

The Flasher ATE supports programming of serial numbers. In order to use the serial number programming feature, the J-Flash project to be used as well as some files on the Flasher ATE (depending on the configuration) need to be configured first.

In general, Flasher ATE supports two ways of programming a serial number into the target:

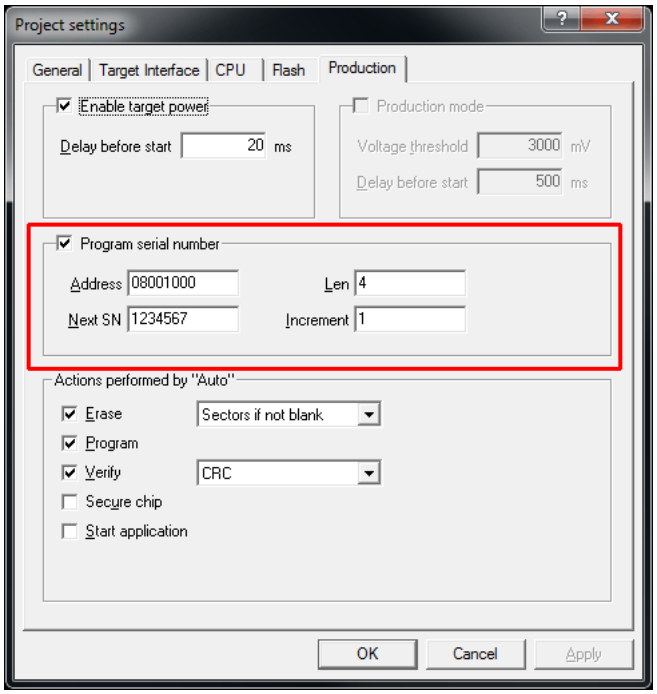
1. Programming continuous serial numbers. Serial number is 1-4 bytes in size. Start serial number, increment, serial number size and address is configured in the J-Flash project.
2. Programming custom serial numbers from a serial number list file. Start line into serial number list file to get next serial number bytes, line increment, serial number size and address is configured in J-Flash project. Serial number list file needs to be specified and created by user.

Some generic information on how to setup Flasher ATE & the J-Flash project for serial number programming are provided below.

> **Note**
>
> Full serial number programming support has been introduced with V4.51d of the J-Flash software and the Flasher firmware that comes with it.

4.1.1 Serial number settings

In order to enable the programming of serial numbers in stand-alone mode, the J-Flash project has to be configured to enable programming a serial number at a specific address. This is done by enabling the **Program serial number** option as shown in the screenshot and table below:



Setting	Meaning
Address	The address the serial number should be programmed at.
Len	The length of the serial number (in bytes) which should be programmed. If no serial number list file is given, J-Flash allows to use a 1-4 byte serial number. In case 8 is selected as length, the serial number and its complementary are programmed at the given address.

Setting	Meaning
	In case a serial number list file is given, Flasher ATE will take the serial number bytes from the list file. If a serial number in the list file does not define all bytes of `Len` the remaining bytes are filled with 0s. No complements etc. are added to the serial number.
`Next SN`	In case no serial number list file is given, `Next SN` is the next serial number which should be programmed. The serial number is always stored in little endian format in the flash memory. In case a serial number list file is given, `Next SN` describes the line of the serial number list file where to read the next serial number bytes from. Flasher ATE starts counting with line 0, so in order to start serial number programming with the first line of the `SNList.txt`, `Next SN` needs to be set to 0.
`Increment`	Specifies by how much `Next SN` is incremented.

4.1.2 Continuous Serial numbers

The Flasher ATE can generate serial numbers. Therefore the project can be configured to use teh serial number feature (see on page 47). The Flasher ATE will use the first serial number for the first programmed device. Then the increment is added to the serial number and this is used for the next programming sequence. The next serial number is stored in the SERIAL.TXT file on each flash module. So the serial number is also power cycle safe. If the file is missing at start up time, the number 0 is used for the first target.

To avoid doubled serial numbers with the Flasher ATE using more than one flash module there are two options:

- Use an increment of the channel number, e.g. if you have 5 flash modules, use an increment of 5 as well as 5 different SERIAL.TXT files at beginning of production.
- Use different serial number areas, e.g. if you have 5 flash modules, use an increment of 1 as well as 5 different SERIAL.TXT files at beginning of production. For the first module, use the range from 1 to 1000, for the second 1001 to 2000 and so on.

The SERIAL.TXT file contains the value `Next SN` in ASCII notation, e.g. `1234` if the next serial number is 1234.

> **Note**
>
> The serial number in SERIAL.TXT will also be incremented serial number programming is disabled, to make sure that for the Flasher ATE logfile there is a reference for which programming cycle passed and which did not. As long as serial number programming has not been enabled in the J-Flash project, Flasher ATE does not merge any serial number data into the image data to be programmed.

4.1.3 Serial number list file

In order to program custom serial numbers which can not be covered by the standard serial number scheme provided by J-Flash (e.g. when programming non-continuous serial numbers or having gaps between the serial numbers), a so called serial number list file needs to be created by the user.

The SERIAL.TXT file needs to contain the values for the serial numbers in ASCII notation. Each line in the file must contain one serial number.

Example

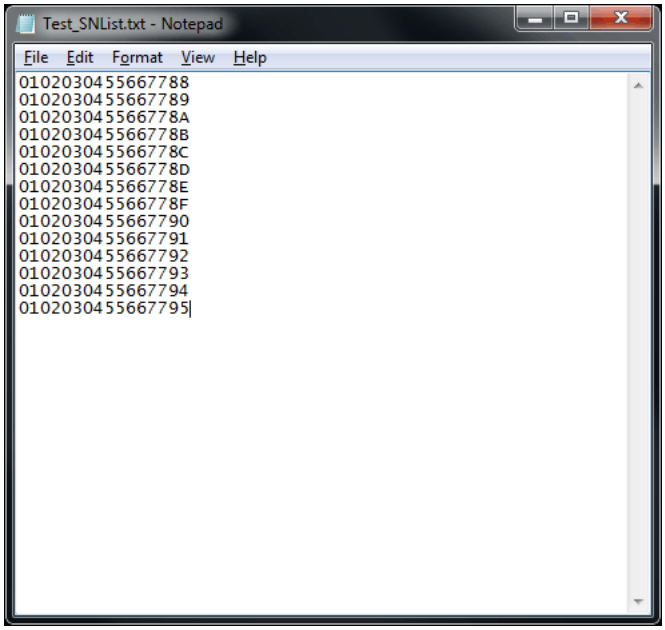
An 8-byte serial number should be programmed at address `0x08000000`.

It should be programmed as follows in the memory:

```
0x08000000: 0x01 0x02 0x03 0x04 0x55 0x66 0x77 0x88
```


The associated serial number list in teh file should look as follows:

```
0102030455667788
```



The number of bytes to read per line is configured via the `Len` option in J-Flash. For more information, please refer to *Serial number settings* on page 47.

Which line Flasher will read at the next programming cycle is configured via the `Next SN` option in J-Flash. For more information, please refer to *Serial number settings* on page 47. In this case, `Next SN` needs to be set to 0, since programming should be started with the serial number bytes defined in the first line of the file.

> **Note**
>
> If the number of bytes specified in a line of the serial number list file is less than the serial number length defined in the project, the remaining bytes are filled with 0s by Flasher ATE.

> **Note**
>
> If the number of bytes specified in a line of the serial number list file is greater than the serial number length defined in the J-Flash project, the remaining bytes will be ignored by Flasher ATE.

4.1.4 Programming process

The Flasher ATE will increment the serial number in SERIAL.TXT by the value defined in `Increment` after each successful programming cycle.

For each programming cycle, the FLASHER.LOG file on the Flasher ATE flash module is updated and contains the value from SERIAL.TXT that has been used for the programming cycle.

> **Note**
>
> The serial number in SERIAL.TXT will also be incremented if serial number programming is disabled, to make sure that for the Flasher ATE logfile there is a reference for which programming cycle passed and which did not. As long as serial number

> programming has not been enabled in the J-Flash project, the Flasher ATE does not merge any serial number data into the image data to be programmed.

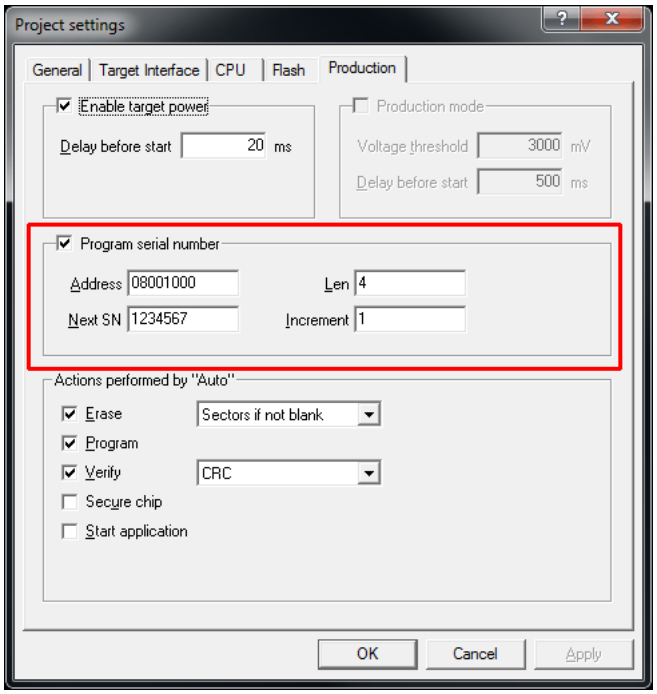
4.1.5 Sample setup

Below, a small example is given on how to setup Flasher ATE for serial number programming. In the following example, 4-byte serial numbers starting at 1234567 (`0x12D687`) shall be programmed at address `0x08001000`.

Defining serial number address, length and start value

In the J-Flash project the following needs to be defined:

- `Address` is `0x08001000`
- `Next SN` is 1234567
- `Increment` is 1
- `Len` is 4 (bytes)



Downloading configuration, data and serial number to the Flasher ATE.

After setting up the rest of the configuration (Target interface etc.) and selecting an appropriate data file, the configuration, data, and serial number file needs to be downloaded into Flasher ATE via FTP client.

4.2 Limiting the number of programming cycles

The Flasher ATE provides a mechanism to limit the number of programming cycles that can be performed in stand-alone mode with the configuration that is stored on the Flasher ATE. To make use of this feature, a file called `Cntdown.txt` needs to be placed on the Flasher ATE module folder. This file simply contains a decimal number (32-bit unsigned integer) that describes how many programming cycles can be performed with the current setup.

> **Note**
>
> The number in the `Cntdown.txt` is only updated on a successful programming cycle. Programming cycles that failed do not affect the `Cntdown.txt`.

4.2.1 Changed fail/error LED indicator behavior

In case a `Cntdown.txt` is found at boot time, the fail/error LED of Flasher behaves different from normal. If the number of programming cycles left is 10 or below, the following will happen:

- The red error/fail LED will be lit for 1 second
- After this, it will blink/toggle x times @ 5 Hz, indicating the number of programming cycles left. (blinking 5 times for 5 cycles left, …)

Chapter 5

Patch data file

This chapter describes how the Flasher ATE can patch data files.

5.1 Patch file support

In stand-alone mode the Flasher ATE supports patch files which allows to patch the content of the data to be programmed. Before starting programming process in stand-alone mode, the Flasher ATE will look for a file named `Patches.txt` being present on the Flasher ATE module. This file includes the patches. If this file is present, the number in `Serial.txt` describes the line number of the Patches.txt that will be used for the current cycle (line counting starts at 0).

Each line in the `Patches.txt` can hold up to 4 patches, where each patch can be up to 32 bytes in length.

Syntax

Each line begins with `<NumPatches>` followed by each patch `<Addr>,<NumBytes>:<Data>` in sequence and separated by commas. So the syntax for `<NumPatches>` = 4 would be as follows:

`<NumPatches>,<Addr>,<NumBytes>:<Data>,<Addr>,<NumBytes>:<Data>,<Addr>,<NumBytes>:<Data>,<Addr>,<NumBytes>:<Data>\r\n`

Find below a table which describes each parameter.

Parameter	Description
`<NumPatches>`	Describes the number of patches in this patch line. Max. value is 4.
`<Addr>`	Describes the address to be patched. Value is expected in hex.
`<NumBytes>`	Number of bytes for the current patch. Max. value is 20h (32 in decimal). Value is expected in hex.
`<Data>`	Describes the data to be patched. <Data> is always expected as 2 hexadecimal characters per byte.

> **Note**
>
> All values are expected in hexadecimal format (hex).
> <Data> section is always preceded by ":", not ",".

Example

Please find below a sample sequence which clarifies the usage of patch files.

Patches.txt, which is located on the Flasher, contains the following line:

3,100025,3:AABBCC,100063,2:DDEE,100078,1:FF

Serial.txt contains a "0" which forces the Flasher to use line 0 from Patches.txt.

After starting the programming cycle, the following data will be patched:

Addr `0x100025`: 3 byte `0xAA 0xBB 0xCC`
Addr `0x100063`: 2 byte `0xDD 0xEE`
Addr `0x100078`: 1 byte `0xFF`

Single patch via RS232 or Telnet

Alternatively, you can start a programming cycle with patch data that is only valid for this one cycle (no need for a Patches.txt file):

Send the `#AUTO PATCH <module> <NumPatches>,<Addr>,<NumBytes>:<Data>`

command via the Flasher ATE ASCII interface. The parameters have the same function as described in the table above.

5.2 FTP server connection

Chapter 6

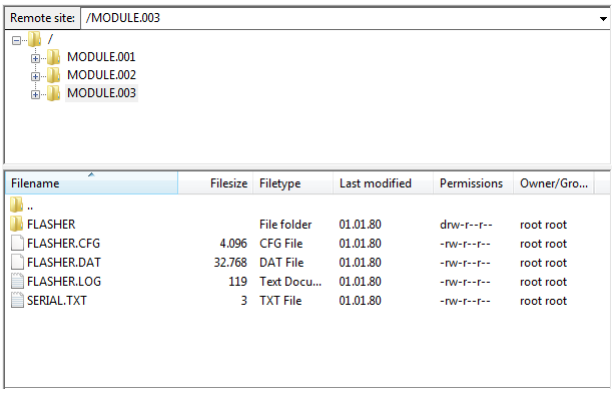
FTP Server

This chapter describes the FTP server features.

6.1 FTP server connection

The FTP server provides easy access to the files on the internal file system. The server supports a maximum of 2 simultaneous connections and works with all common FTP clients.

For the Flasher ATE, the FTP server is the only way to access files on the modules. Here the root directory is a virtual directory and cannot be written to. It contains a subdirectory for each module.
The module next to the mainboard is the first one and is mounted as MODULE.001 and the last module is mounted as Module.010.



The FTP server allows you to upload or download the target configuration and data files. The Flasher ATE setup files can also be uploaded or downloaded to the module folders.

The Flasher ATE writes a log file for executed operations. This can be found in the modules folder and downloaded from there.

> **Note**
>
> The file system on the Flasher ATE supports only 8.3 file names. Meaning 8 characters for the name and 3 characters for the file extension.

The IP setup is described here: *Setting up the IP interface* on page 20.

6.1.1 Access data

Anonymous access to the FTP server is limited to read-only access to the file system.
For write access, special login credentials have to be used:

```
Login: admin
Password: 1234
```

> **Note**
>
> The access data for read/write access can not be modified and it is intended to be used only as a convenience feature to avoid unintended modification of the Flasher's file system. It is not meant as a security feature.

Chapter 7

Web server

This chapter describes the web server features.

7.1 Web server features

The Flasher ATE comes with a built-in web server, which provides a web interface for information and network configuration. For the network, the IP address settings can be changed and a nick name can be assigned to the device.

Additionally, the web interface provides information about the status of the integrated operating system, the IP stack and the target hardware. The Flasher ATE's web interface furthermore allows monitoring of the individual modules.

The IP setup is described here: *Setting up the IP interface* on page 20.

Chapter 8

Remote control

This chapter describes how to control Flasher via the 9-pin serial interface connector or via the integrated Telnet interface.

8.1 Overview

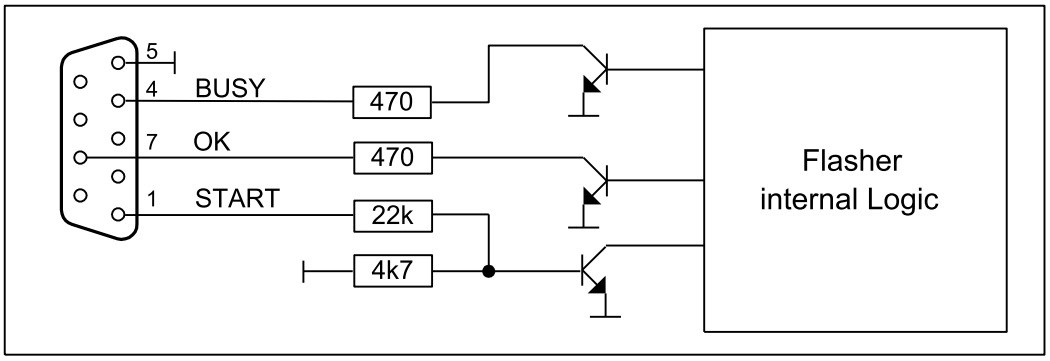
There are three ways to remote control the Flasher ATE operation:

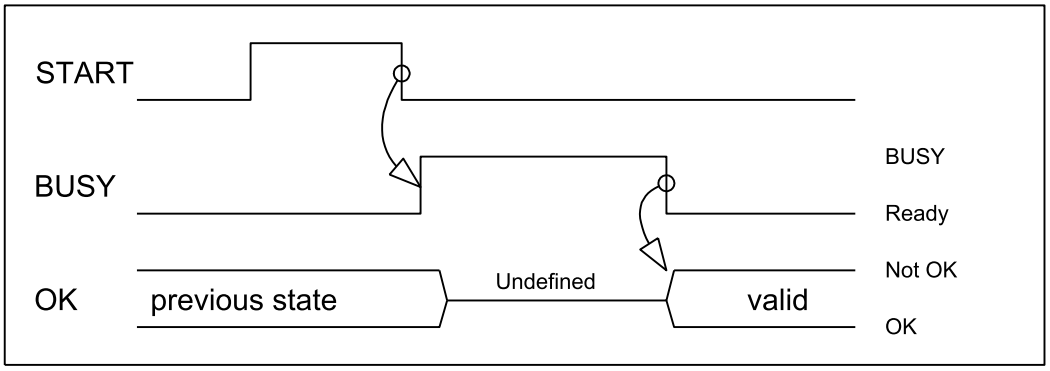
- Via Handshake lines: 3 lines on the serial interface are used:
  1 line is an input and can be used to start operation,
  2 lines are outputs and serve as busy and status signals.
- Terminal communication via RS232.
- Terminal communication via Telnet.

8.2 Handshake control

The Flasher ATE can be remote-controlled by automated testers without the need of a connection to a PC. Therefore the Flasher ATE is equipped with additional hardware control functions, which are connected to the SUBD9 male connector, normally used as RS232 interface to PC.

The following diagrams show the internal remote control circuitry of Flasher:





Pin No.	Function	Description
1	START	A positive pulse of any voltage between 5 and 30V with duration of min. 30 ms starts “Auto” function (Clear / Program / Verify) on falling edge of pulse. The behavior of the “Auto” function depends on the project settings, chosen in J-Flash at the **Production** tab.
4	BUSY	As soon as the “Auto” function is started, BUSY becomes active, which means that transistor is switched OFF.
5	GND	Common Signal ground.
7	OK	This output reflects result of last action. It is valid after BUSY turned back to passive state. The output transistor is switched ON to reflect OK state.

Note

As the Flasher ATE is a modular system, using the handshake remote control START always triggers the “Auto” function of every connected module. The BUSY line is signaled as long as any module is still busy and the OK line only reports “OK” in case of every module has successfully completed the operation. We recommend using the

ASCII command interface, described in the next chapter, for the Flasher ATE as it gives better remote control capabilities.

8.3 ASCII command interface

8.3.1 Introduction

Once set up using J-Flash, the Flasher ATE can be driven by any application or just a simple terminal using ASCII commands.

Every known command is acknowledged by the Flasher ATE and then executed. After command execution, the Flasher ATE sends an ASCII reply message.

> **Note**
>
> There are situations where the execution of a known command is rejected with #NACK:ERRxxx if Flasher ATE is currently busy and the received command is not allowed to be sent while Flasher ATE is busy

8.3.2 General command and reply message format

- Any ASCII command has to start with the start delimiter #.
- Any ASCII command has to end with simple carriage return ('\r', ASCII code 13).
- Commands can be sent upper or lower case.

8.3.3 General usage

Reply messages must be considered in each case. In general, a new command must not be sent before a reply for the last one has been received. The least the "#ACK" needs to be received by the controlling application before sending a new command for a flash module not yet executing a command.

For the Flasher ATE, all commands triggering a flash programming function (`#AUTO`, `#CANCEL`, `#ERASE`, `#PROGRAM`, `#VERIFY`) may be used for other modules, before the current operation has been finished. Please note that in this case the overall finish indicator "#DONE" will be sent when all commands have been executed.

When a flash programming function has finished, the debug logic of the MCU is disabled (power down) and the target interface of the module is switched off (tristated).

8.3.4 Settings for ASCII interface via RS232

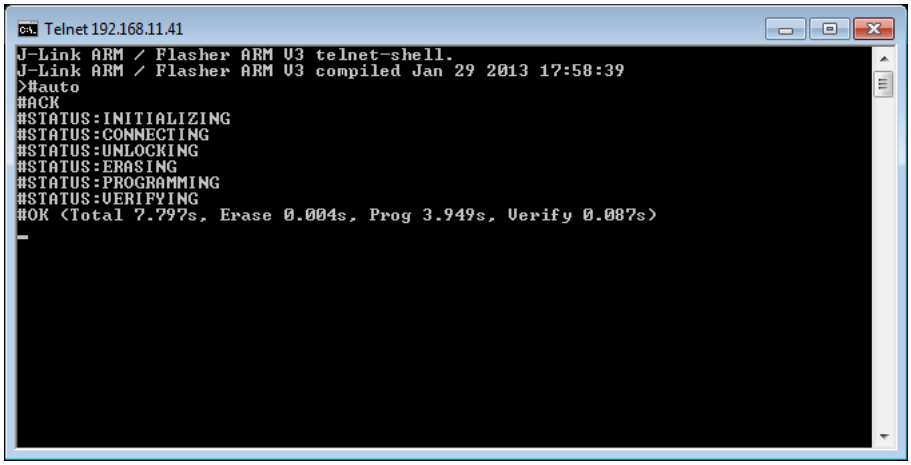
Flasher is driven via a RS232 serial port with the following initial interface settings:

- 9600 baud
- 8 data bits
- no parity
- 1 stop bit

The baud rate can be changed by using the `#BAUDRATE` command.

8.3.5 Settings for ASCII interface via Telnet

A client application can connect to the Flasher ATE via Telnet on port 23. Find below a screenshot of Flasher which is remote controlled via Telnet:



Some additional data are transferred via an additional telnet connection (target channel) for each module. These data may be helpful for the project setup. In the normal operation these channels are not required.

8.3.6 Commands and replies

The table below gives an overview about the commands which are supported by the current version of the Flasher ATE firmware. Click on the names for a detailed description:

Commands to the Flasher ATE
#BAUDRATE<Baudrate>
#AUTO <Module1>[,<Module2>]
#AUTO NOPATCH <Module1>[,<Module2>]
#AUTO PATCH <Module1>[,<Module2>] [number of patches],[address],[number of patched bytes]:[data bytes]
#CANCEL <Module1>[,<Module2>]
#ERASE <Module1>[,<Module2>]
#FFORMAT <Module1>[,<Module2>]
#FWVERSION
#FWVERSIONMOD <Module1>[,<Module2>]
#IPCONFIG
#POWERON <Module1>[,<Module2>], [PowerSource, Discharge Mode]
#POWEROFF <Module1>[,<Module2>]
#PROGRAM <Module1>[,<Module2>]
#PROTVER
#RESULT <Module1>[,<Module2>]
#RTTON <Module1>[,<Module2>] [Channel],[RTT Control Block Address], [Down Buffers], [Up Buffers]
#RTTOFF <Module1>[,<Module2>]
#SELECT <Module1>[,<Module2>] <Filename>
#SETVTREF <Module1>[,<Module2>] <vtref voltage in mV>
#START <Module1>[,<Module2>]
#STATUS <Module1>[,<Module2>]
#VERIFY <Module1>[,<Module2>]
#SELMODULE <Module1>[,<Module2>]
#SERIAL
#SERIALMOD <Module1>[,<Module2>]
#SETVTREF <Module1>[,<Module2>] [voltage]
#TERMINAL <Module1>[,<Module2>] <Baudrate>,<Databits>,<Parity>,<Stopbits>

Replies from the Flasher ATE
#ACK
#NACK
#OK
#OK:<NumBytes>:<Data>
#OK:<Data>
#OK:<Module>:<Data>
#RESULT:<Module>:
#DONE
#ERRxxx

8.3.6.1 Commands to the Flasher

8.3.6.1.1 Command #AUTO

The #AUTO command behaves exactly as the external remote control input.

Usually, the following command sequence will be performed when receiving the #AUTO command:

- The Flasher ATE erases the target CPU (if not blank)
- The Flasher ATE programs the target CPU
- The Flasher ATE verifies the target CPU

Depending on the settings chosen in the **Production** tab in the J-Flash tool, this sequence can differ from the one shown above.

Command structure:

```
#AUTO [module1][, module2][, module3]
```

Alternatively the modules can be replaced:

- *, will execute the auto command using all modules selected by the latest executed selmodule command.
- all, will execute the auto command using all modules which can be detected.

Results of the Flasher ATE:

Result	Meaning
#OK	successfully done
#ERRxxx	if any error occurred during operation. xxx represents the error code, normally replied to Flasher ATE PC program. #ERRxxx message may be followed by an additional error text.

During execution of the #AUTO command, Flasher ATE automatically sends “status” messages via the terminal connection to reflect the state of execution.

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#AUTO 1		
	#ACK	
		#STATUS:INITIALIZING
		#STATUS:CONNECTING
		#STATUS:UNLOCKING
		#STATUS:ERASING
		#STATUS:PROGRAMMING
		#STATUS:VERIFYING
		#OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)
		#STATUS:READY
	#RESULT:1:OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)	

8.3.6.1.2 Command #AUTO NOPATCH

The #AUTO NOPATCH command allows to ignore an existing patch file for the programming.

The background about this is that the 3-wire handshake protocol shall be able to patch data files, too. So the default behavior of the #auto command is that an existing patch file (patch.txt in the module folder) is applied to a data if the #auto command is executed.

Flasher ATE responds with

- #OK if no error occurred
- #ERRxxx if any error occurred during operation. xxx represents the error code, normally replied to Flasher PC program. The #ERRxxx message may be followed by an additional error text.

For further information about the usage of the #AUTO PATCH command please refer to *Patch file support* on page 54.

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#AUTO NOPATCH 1		
	#ACK	
		#STATUS:INITIALIZING
		#STATUS:CONNECTING
		#STATUS:UNLOCKING
		#STATUS:ERASING
		#STATUS:PROGRAMMING
		#STATUS:VERIFYING
		#OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)
		#STATUS:READY
	#RESULT:1:OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)	

8.3.6.1.3 Command #AUTO PATCH

The #AUTO PATCH command allows patching of the content of the data to be programmed.

Flasher ATE responds with

- #OK if no error occurred
- #ERRxxx if any error occurred during operation. xxx represents the error code, normally replied to Flasher PC program. The #ERRxxx message may be followed by an additional error text.

For further information about the usage of the #AUTO PATCH command please refer to *Patch file support* on page 54.

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#AUTO PATCH 1 1,0,8:0011223344556677		
	#ACK	
		#STATUS:INITIALIZING
		#STATUS:CONNECTING
		#STATUS:UNLOCKING
		#STATUS:ERASING
		#STATUS:PROGRAMMING
		#STATUS:VERIFYING
		#OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)
		#STATUS:READY
	#RESULT:1:OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)	

8.3.6.1.4 Command "#BAUDRATE"

This command can be sent in order to change the baud rate of the Flasher ATE ’s RS232 interface used for communication. <Baudrate> is expected in decimal format. The Flasher ATE supports baud rates form 1.200 to 115.200 bit/s.

Command structure:

```
#Baudrate [Baudrate]
```

Results of the Flasher ATE:

Result	Meaning
#OK	successfully done
#ERR255: Invalid parameters	the baud rate parameter is invalid, e.g. contains not parseable characters
#ERR255: Baudrate is not supported	the selected baud rate is not supported by the Flasher ATE, e.g. it is to fast or slow.

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#BAUDRATE 115200		
	#ACK	
	#OK	

Note

After sending the #BAUDRATE command you will first have to wait until the Flasher ATE responds with the #OK message. It is recommended wait further 5ms before sending the next command with the new baud rate in order to give the Flasher ATE the time to change the baud rate.

8.3.6.1.5 Command #CANCEL

This command can be sent to abort a running program. It may take a while until the current program is actually canceled.

Command structure:

```
#Cancel [module1][, module2][, module3]
```

Results of the Flasher ATE:

Result	Meaning
`#ERR007:CANCELED.`	successfully canceled the operation

Example:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
`#CANCEL 1`		
	`#ERR007:CANCELED`	

8.3.6.1.6 Command "#ERASE"

This command can be sent to erase all selected target flash sectors.

Command structure:

`#ERASE [module1][,module2][,module3]`

Results of the Flasher ATE:

Result	Meaning
#OK	successfully done
#ERRxxx: TEXT	error message with text

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#ERASE 1		
	#ACK	
		#STATUS:INITIALIZING
		#STATUS:CONNECTING
		#STATUS:UNLOCKING
		#STATUS:ERASING
		#OK (Total 0.362s, Erase 0.252s)
		#STATUS:READY
	#RESULT:1:OK (Total 0.362s, Erase 0.252s)	

8.3.6.1.7 #FFORMAT

This command formats the file system on the specified module.

Command structure:

```
#FFORMAT [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#FFORMAT 1		
	#ACK	
	#RESULT:1:OK	
	#DONE	

8.3.6.1.8 #FWVERSION

This command returns the firmware version of the mainboard.

Command structure:

```
#FWVERSION
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#FWVERSION		
	#ACK	
	#OK:1:1.04d	
	#DONE	

8.3.6.1.9 #FWVERSIONMOD

This command returns the firmware of a module.

Command structure:

```
#FWVERSIONMOD [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#FWVSERIONMOD 1,2,3		
	#ACK	
	#OK:1:1.04d	
	#OK:2:1.04d	
	#OK:3:1.04d	
	#DONE	

8.3.6.1.10 #IPCONFIG

This command returns the currently used IP configuration.

Command structure:

`#IPCONFIG`

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
`#IPCONFIG`		
	`#ACK`	
	`#RESULT:IP address:192.168.1.111`	
	`#RESULT:subnet mask:255.255.0.0`	
	`#RESULT:Gateway:192.168.1.1`	
	`#RESULT:IP mode:automatic(DHCP) assigned`	
	`#DONE`	

IP mode can be `automatic(DHCP) assigned` or `manual assigned`.

8.3.6.1.11 #PROGRAM

This command can be used instead of #AUTO to program a target without erasing the target before programming and without performing a final verification.

Command structure:

```
#PROGRAM [module1][,module2][,module3]
```

Flasher ATE will reply the following sequence of messages:

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#ERASE 1		
	#ACK	
		#STATUS:INITIALIZING
		#STATUS:CONNECTING
		#STATUS:UNLOCKING
		#STATUS:PROGRAMMING
		#OK (Total 9.963s, Prog 9.183s)
		#STATUS:READY
	#RESULT:1:OK (Total 9.963s, Prog 9.183s)	

8.3.6.1.12 #POWERON

This command can be used to turn on the target power without any erase, program or verify action.

Command structure:

```
#POWERON [module1][,module2][,module3] [Power Source],[Discharge]
```

The power on command expects the following parameters:

Parameter	Meaning
Power Source	0 = internal power, 1 = VTgt
Discharge	1 = discharge target when turning the power off (needs to be set with the power on command), 0 = no discharge

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
`#POWERON 1,2,3 1,0`		
	`#ACK`	
	`#DONE`	

The connection of the power supply is described in the chapter *Target power supply* on page 97.

8.3.6.1.13 #POWEROFF

This command can be used to turn on or off the target power without any erase, program or verify action.

Command structure:

```
#POWEROFF [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#POWEROFF 1,2,3		
	#ACK	
	#DONE	

The connection of the power supply is described in the chapter *Target power supply* on page 97.

8.3.6.1.14 #RESULT

This command can be sent any time, even during other command execution. Flasher responds with the last result of the previously executed command.

Command structure:

```
#RESULT [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#RESULT 1,2,3		
	#ACK	
	#RESULT:1:OK ((Total 2.216s, Erase 0.126s, Prog 1.231s, Verify 0.144s)	
	#RESULT:2:OK ((Total 2.216s, Erase 0.126s, Prog 1.231s, Verify 0.144s)	
	#RESULT:3:OK ((Total 2.216s, Erase 0.126s, Prog 1.231s, Verify 0.144s)	
	#DONE	

8.3.6.1.15 #RTTON

This command turns on the RTT connection for the given modules.

Command structure:

```
#RTTON [module1][,module2][,module3] [RTT channel],[RTT control block address], [Number Down Buffers], [Number Up Buffers]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#RTTON 1,2,3 0,0x20001000,3,3		
	#ACK	
	#DONE	

The target data are transferred via an additional telnet connection. The telnet connection for module 1 is available on port 41 up to module 10 on port 50.

8.3.6.1.16 #RTTOFF

This command turns off the RTT connection for the given modules.

Command structure:

```
#RTTOFF [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#RTTOFF 1,2,3		
	#ACK	
	#DONE	

8.3.6.1.17 #SELECT

The #SELECT command is used to select a specific configuration and data file pair which should be used by the Flasher ATE to program the target.

Command structure:

```
#select [module1][,module2][,module3] “[Project Name]”
```

The select command expects the following parameters:

Parameter	Meaning
Project Name	The [Project Name] specifies the name of file pair without extensions (.CFG and .DAT) on the Flasher ATE modules which should be selected. Flasher saves the selected configuration and data file in the FLASHER.INI file. So this selection is remembered even after power-cycling the Flasher ATE.

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#SELECT 1,2,3 “emPower”		
	#ACK	
	#OK	

8.3.6.1.18 #SERIAL

The #SERIAL command is used query the serial number of the mainboard.

Command structure:

```
#serial
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#serial		
	#ACK	
	#RESULT:871012345	
	#DONE	

8.3.6.1.19 #SERIALMOD

The `#SERIALMOD` command is used query the serial numbers of the flash boards.

Command structure:

```
#SERIALMOD [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
`#serialmod 1,2,3`		
	`#ACK`	
	`#RESULT:1:891010001`	
	`#RESULT:2:891010004`	
	`#RESULT:3:891010007`	
	`#DONE`	

8.3.6.1.20 #START

This command can be sent to start the application using the method configured in the J-Flash project.

Command structure:

```
#START [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#START 1,2,3		
	#ACK	
		#STATUS:INITIALIZING
		#STATUS:CONNECTING
		#OK (Total 0.083s)
		#STATUS:READY
	#RESULT:1:OK (Total 0.083s)	
	#RESULT:3:OK (Total 0.082s)	
	#RESULT:3:OK (Total 0.084s)	
	#OK	

8.3.6.1.21 #STATUS

This command can be sent any time, even during other command execution. Flasher ATE responds with its current state. All defined state messages are described under *Replies from Flasher ATE* on page 92.

Command structure:

```
#STATUS [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#STATUS 1,2,3		
	#ACK	
	#STATUS:1:READY	
	#RESULT:3:CONNECTING	
	#RESULT:3:PROGRAMMING	
	#OK	

8.3.6.1.22 #VERIFY

This command can be used to verify the target flash content against the data stored in Flasher ATE.

Command structure:

```
#VERIFY [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
#VERIFY 1,2,3		
	#ACK	
		STATUS:INITIALIZING}
		STATUS:CONNECTING
		STATUS:VERIFYING
		#OK (Total 0.129s)
		STATUS:READY
	#RESULT:1:OK (Total 0.206s, Verify 0.129s)	
	#RESULT:2:OK (Total 0.210s, Verify 0.131s)	
	#RESULT:3:OK (Total 0.207s, Verify 0.128s)	
	#OK	

8.3.6.1.23 #SETVTREF

This command can be used to set a fix voltage for I/O pins of the target interface.

Command structure:

`#SETVTREF [module1][,module2][,module3] [voltage level]`

The SETVTREF command expects the following parameters:

Parameter	Meaning
voltage level	The IO voltage level for the target interface in mV.

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
`#SETVTREF 1,2,3 3300`		
	#ACK	
	#OK	

8.3.6.1.24 #SELMODULE

This command is used to select one or more modules on a Flasher ATE system. The module numbers are separated by a comma. If all modules shall be selected, the keyword "all" can be used (`#SELMODULE ALL`) instead of a list with all module numbers.

Command structure:

```
#SELMODULE [module1][,module2][,module3]
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
`#SELMODULE 1,2,3`		
	`#ACK`	
	`#SELECTED:1,2,3`	

8.3.6.1.25 #TERMINAL

This command enables the UART transceiver. The command is completed by the parameters:

Command structure:

```
#TERMINAL [module1][,module2][,module3] [Baudrate],[Data bits],[Parity],[Stop bits]
```

The `TERMINAL` command expects the following parameters:

Parameter	Meaning
Baudrate	The baudrate of the UART which can be between 600 and 115,200 bits per second.
Data bits	The data bits of an UART byte. The data bits parameter supports currently only 8 bit.
Parity	Parity can be 1) N (no parity), 2) E (even parity) or 3) O (odd parity).
Stop bits	The stop bits parameter supports currently only 1 stop bit.

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
`#TERMINAL 1,2,3 115200,8,E,1`		
	#ACK	
	#0K	

The target data are transferred via an additional telnet connection. The telnet connection for module 1 is available on port 41 up to module 10 on port 50.

If you want to turn off the UART terminal mode use the parameter off.

Command structure:

```
#TERMINAL [module1][,module2][,module3] off
```

Example sequence:

Command send to the Flasher ATE	Reply on the main channel (RS232 or telnet connection)	Reply on the target channel (telnet connection to flash module)
`#TERMINAL 1,2,3 off`		
	#ACK	
	#0K	

> **Note**
>
> The terminal feature uses the following pins: Pin 5 = Flasher-Tx (out), Pin 17 = Flasher-Rx (in).

8.3.6.2 Replies from Flasher ATE

The reply messages from Flasher ATE follow the same data format as commands. Any reply message starts with ASCII start delimiter #, ends with simple carriage return (ASCII code 13) and is sent in uppercase. In contrast to commands, replies can be followed by a descriptive message, which gives more detailed information about the reply. This description is sent in mixed case. The #OK reply, for example, is such a reply. It is followed by a string containing information about the performance time needed for the operations:

```
#OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)
```

The following reply messages from Flasher ATE are defined:

8.3.6.2.1 #ACK

Flasher replies with #ACK message on reception of any defined command before the command itself is executed.

8.3.6.2.2 #NACK

Flasher replies with #NACK, if an undefined command was received.

8.3.6.2.3 #OK

Flasher replies with #OK, if a command other than #STATUS or #RESULT was executed and ended with no error.

8.3.6.2.4 #OK:<Data>

Flasher replies with #OK:<Len>:<Data> if a #FREAD command was executed. <NumBytes> is the number of bytes which could be read. This value may differ from the number of requested bytes, for example if more bytes than available, were requested. <NumBytes> and <Data> are send in hexadecimal format (for <Data>: two hexadecimal characters per byte).

8.3.6.2.5 #STATUS:<data>

The Flasher ATE replies with its current state.

The following status messages are currently defined:

Message	Description
#STATUS:READY	Flasher is ready to receive a new command.
#STATUS:CONNECTING	Flasher initializes connection to target CPU.
#STATUS:INITIALIZING	Flasher performs self check and internal init.
#STATUS:UNLOCKING	Unlocking flash sectors.
#STATUS:ERASING	Flasher is erasing the flash of the target device.
#STATUS:PROGRAMMING	Flasher is programming the flash of the target device.
#STATUS:VERIFYING	Flasher verifies the programmed flash contents.

8.3.6.2.6 #RESULT:<Module>:<data>

The Flasher ATE reports the result of an operation on a specific module. If the operation has been completed successfully, it will report the outcome with a single message of this type followed by the last result of the operation.

8.3.6.2.7 #DONE

For the Flasher ATE, getting a result from a module does not necessarily mean, the Flasher ATE is in idle state. Therefore, this message is being sent, once all operations are finished and all modules are back in idle state.

A typical sequence for using the Flasher ATE is as follows:

```
J-Link / Flasher ATE Mainboard V1 telnet-shell.
J-Link / Flasher ATE Mainboard V1 compiled Mar 15 2018 12:28:43
#SELMODULE 1,2
#ACK
#SELECTED:1,2
#AUTO *
#ACK
#RESULT:1:#ERR255:Error while flashing
#RESULT:2:#OK (Total 2.653s, Erase 0.327s, Prog 1.960s, Verify 0.234s)
#DONE
```

8.3.6.2.8 #ERRxxx <Data>

If any command other than #STATUS or #RESULT was terminated with an error, Flasher ATE cancels the command and replies with an error message instead of #OK message.

Some error codes may be followed by colon and an additional error text.

For example:

```
#ERR007:CANCELED.
```

The error code numbers are described in the following table:

Message	Description
#ERR007	Flasher received #CANCEL command and has canceled the current operation.
#ERR008	Flasher is already busy with execution of previous command.
#ERR009	Failed to allocate memory.
#ERR010	Failed to open file.
#ERR011	Failed to read file.
#ERR012	Failed to write file.
#ERR013	Failed to delete file.
#ERR098	Failed to delete file.
#ERR098	Could not allocate memory for device specific algorithm.
#ERR099	Device specific algorithm is not yet supported by this firmware version. Please check for an update.;
#ERR101	Could not find device programming algorithm.
#ERR102	Could not open the data file.
#ERR255	Undefined error occurred. This reply is followed by an error string.

Chapter 9

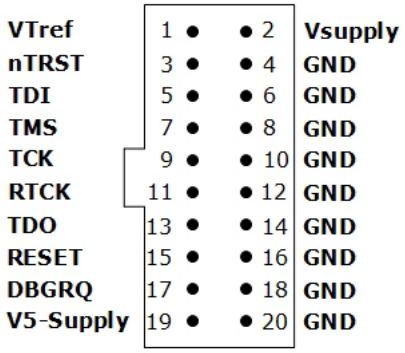
Hardware

This chapter gives an overview about Flasher ATE specific hardware details, such as the pinouts and available adapters.

9.1 Flasher ARM 20-pin JTAG/SWD Connector

Flasher has a JTAG connector compatible with ARM's Multi-ICE. The JTAG connector is a 20 way Insulation Displacement Connector (IDC) keyed box header (2.54mm male) that mates with IDC sockets mounted on a ribbon cable.

9.1.1 Pinout JTAG



The following table lists the Flasher JTAG pinout.

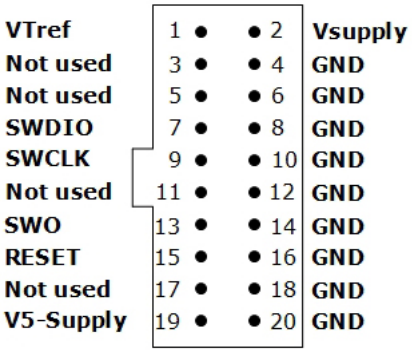
PIN	SIGNAL	TYPE	Description
1	VTref	Input	This is the target reference voltage. It is used to check if the target has power, to create the logic-level reference for the input comparators and to control the output logic levels to the target. It is normally fed from Vdd of the target board and must not have a series resistor.
2	Vsupply	NC	This pin is not connected to Flasher ARM. It is reserved for compatibility with other equipment. Connect to Vdd or leave open in target system.
3	nTRST	Output	JTAG Reset. Output from Flasher ARM to the Reset signal of the target JTAG port. Typically connected to nTRST of the target CPU. This pin is normally pulled HIGH on the target to avoid unintentional resets when there is no connection.
5	TDI	Output	JTAG data input of target CPU. It is recommended that this pin is pulled to a defined state on the target board. Typically connected to TDI of target CPU.
7	TMS	Output	JTAG mode set input of target CPU. This pin should be pulled up on the target. Typically connected to TMS of target CPU.
9	TCK	Output	JTAG clock signal to target CPU. It is recommended that this pin is pulled to a defined state of the target board. Typically connected to TCK of target CPU.
11	RTCK	Input	Return test clock signal from the target. Some targets must synchronize the JTAG inputs to internal clocks. To assist in meeting this requirement, you can use a returned, and re-timed, TCK to dynamically control the TCK rate. Flasher ARM supports adaptive clocking, which waits for TCK changes to be echoed correctly before making further changes. Connect to RTCK if available, otherwise to GND.
13	TDO	Input	JTAG data output from target CPU. Typically connected to TDO of target CPU.
15	RESET	I/O	Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called "nRST", "nRESET" or "RESET".

PIN	SIGNAL	TYPE	Description
17	DBGRQ	NC	This pin is not connected in Flasher ARM. It is reserved for compatibility with other equipment to be used as a debug request signal to the target system. Typically connected to DBGRQ if available, otherwise left open.
19	5V-Target supply	Output	This pin is used to supply power to some eval boards. Typically left open on target hardware.

Pins 4, 6, 8, 10, 12, 14, 16, 18, 20 are GND pins connected to GND in Flasher ARM. They should also be connected to GND in the target system.

9.1.2 Pinout SWD

The 20-pin connector of Flasher is also compatible to ARM's Serial Wire Debug (SWD) interface.



The following table lists the Flasher ATE SWD pinout.

PIN	SIGNAL	TYPE	Description
1	VTref	Input	This is the target reference voltage. It is used to check if the target has power, to create the logic-level reference for the input comparators and to control the output logic levels to the target. It is normally fed from Vdd of the target board and must not have a series resistor.
2	Vsupply	NC	This pin is not connected in Flasher ATE. It is reserved for compatibility with other equipment. Connect to Vdd or leave open in target system.
3	Not Used	NC	This pin is not used by Flasher ATE. If the device may also be accessed via JTAG, this pin may be connected to nTRST, otherwise leave open.
5	Not used	NC	This pin is not used by Flasher ATE. If the device may also be accessed via JTAG, this pin may be connected to TDI, otherwise leave open.
7	SWDIO	I/O	Single bi-directional data pin.
9	SWCLK	Output	Clock signal to target CPU. It is recommended that this pin is pulled to a defined state of the target board. Typically connected to TCK of target CPU.
11	Not used	NC	This pin is not used by Flasher ATE. This pin is not used by Flasher ATE when operating in SWD mode. If the device may also be accessed via JTAG, this pin may be connected to RTCK, otherwise leave open.
13	SWO	Output	Serial Wire Output trace port. (Optional, not required for SWD communication.)

PIN	SIGNAL	TYPE	Description
15	RESET	I/O	Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called “nRST”, “nRESET” or “RESET”.
17	Not used	NC	This pin is not connected in Flasher ATE.
19	5V-Target supply	Output	This pin is used to supply power to some eval boards. Typically left open on target hardware.

Pins 4, 6, 8, 10, 12, 14, 16, 18, 20 are GND pins connected to GND in Flasher ATE. They should also be connected to GND in the target system.

9.1.3 Target power supply

Pin 19 of the connector can be used to supply power to the target hardware. Supply voltage is 5V, max. current is 400mA. The output current is monitored and protected against overload and short-circuit.

Power can be controlled via the *ASCII command interface* on page 64 or by enabing power supply in the project configuration.

9.2 Target board design

We strongly advise following the recommendations given by the chip manufacturer. These recommendations are normally in line with the recommendations. Please refer to the the appropriate tables depending on the core:

- *Pinout JTAG* on page 95
- *Pinout SWD* on page 96

In case of doubt you should follow the recommendations given by the semiconductor manufacturer.

9.2.1 Pull-up/pull-down resistors

Unless otherwise specified by developer’s manual, pull-ups/pull-downs are recommended to be between 2.2 kOhms and 47 kOhms.

9.2.2 RESET, nTRST

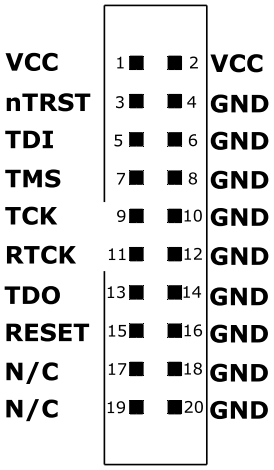
The debug logic is reset independently from the CPU core with nTRST. For the core to operate correctly it is essential that both signals are asserted after power-up.

The advantage of having separate connection to the two reset signals is that it allows the developer performing software debug to setup breakpoints, which are retained by the debug logic even when the core is reset. (For example, at the reset vector address, to allow the code to be single-stepped as soon as it comes out of reset). This can be particularly useful when first trying to bring up a board with a new ASIC.

9.3 Adapters

9.3.1 JTAG Isolator

The JTAG Isolator can be connected between Flasher ATE and JTAG adapter, to provide electrical isolation. This is essential when the development tools are not connected to the same ground as the application. For more information about the JTAG Isolator, please refer to *J-Link JTAG Isolator User Manual* (UM08010) which can be downloaded from our website.



9.3.1.1 Pinout

The following table shows the target-side pinout of the JTAG Isolator adapter.

Pin	Signal	Type	Description
1	VCC	Output	The target side of the isolator draws power over this pin.
2	VCC	Output	The target side of the isolator draws power over this pin.
3	nTRST	Output	JTAG Reset. Output from Flasher ATE to the Reset signal of the target JTAG port. Typically connected to nTRST of the target CPU. This pin is normally pulled HIGH on the target to avoid unintentional resets when there is no connection.
5	TDI	Output	JTAG data input of target CPU. It is recommended that this pin is pulled to a defined state on the target board. Typically connected to TDI of target CPU.
7	TMS	Output	JTAG mode set input of target CPU. This pin should be pulled up on the target. Typically connected to TMS of target CPU.
9	TCK	Output	JTAG clock signal to target CPU. It is recommended that this pin is pulled to a defined state of the target board. Typically connected to TCK of target CPU.
11	RTCK	Input	Return test clock signal from the target. Some targets must synchronize the JTAG inputs to internal clocks. To assist in meeting this requirement, you can use a returned, and re-timed, TCK to dynamically control the TCK rate.
13	TDO	Input	JTAG data output from target CPU. Typically connected to TDO of target CPU.
15	RESET	I/O	Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called “nRST”, “nRESET” or “RESET”.
17	N/C	N/C	This pin is not connected on the target side of the isolator.
19	N/C	N/C	This pin is not connected on the target side of the isolator.

Pins 4, 6, 8, 10, 12, 14, 16, 18, 20 are connected to GND.

9.3.2 J-Link Needle Adapter

To connect to the Flasher ATE via programming interface the *J-Link Needle Adapter* is recommended.

Why to choose the J-Link Needle Adapter:

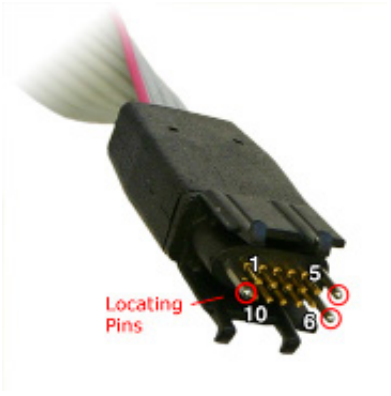
1. No additional connector required on your PCB
2. Very small footprint
3. High reliability spring pins for secure connections
4. Designed with 3 locating pins, so the adapter can not be connected the wrong way
5. No external power supply required! The J-Link Needle Adapter comes with the option to power the target hardware via J-Link.

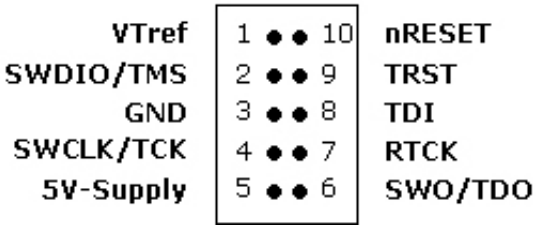
These features make the J-Link Needle Adapter the perfect solution for production purposes.

The pinout of the J-Link Needle Adapter is based on the pinout of the needle adapter by Tag-Connect. Please note, that both pinouts are not identical since the J-Link Needle Adapter comes with a 5V-supply pin.

As you can see on the image below, the three locating pins ensure, that the adapter cannot be connected to the PCB the wrong way.

Moreover, the two “legs” on each side of the connector guarantee a stable and secure contact between pins and the PCB.



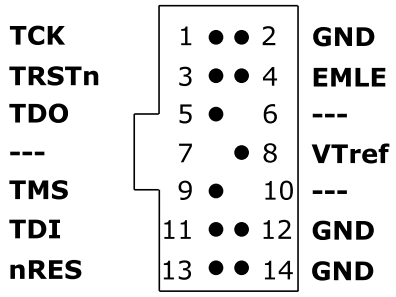


The J-Link Needle Adapter can be connected to J-Link via the *20-pin 0.1” JTAG to a 10-pin needle connector*.

9.3.3 Flasher RX 14-pin Adapter

Flasher ATE itself has a 20-pin JTAG connector mounted but comes with a 14-pin adapter for Renesas RX devices. This adapter also enables Flasher ATE to optionally power the connected target hardware. On the adapter there is a jumper which allows selection between

3.3V and 5V supply target voltage supply. The target is supplied via the VTref connection when the supply option is jumpered.



The following table lists the Flasher RX 14-pin JTAG pinout.

Pin	Signal	Type	Description
1	TCK	Output	JTAG clock signal to target CPU. It is recommended that this pin is pulled to a defined state on the target board. Typically connected to TCK on target CPU.
3	TRSTn	Output	JTAG Reset. Output from Flasher ATE to the Reset signal of the target JTAG port. Typically connected to nTRST of the target CPU. This pin is normally pulled HIGH on the target to avoid unintentional resets when there is no connection.
4	EMLE	Output	Pin for the on-chip emulator enable signal. When the on-chip emulator is used, this pin should be driven high. When not used, it should be driven low. Pulled HIGH to VTref via 1k pull-up resistor on 14-pin adapter.
5	TDO	Input	JTAG data output from target CPU. Typically connected to TDO on target CPU.
6	—	NC	This pin is not connected to Flasher ATE.
7	—	NC	This pin is not connected to Flasher ATE.
8	VTref	Input	This is the target reference voltage. It is used to check if the target has power, to create the logic-level reference for the input comparators and to control the output logic levels to the target. It is normally fed from Vdd of the target board and must not have a series resistor.
9	TMS	Output	JTAG mode set input of target CPU. This pin should be pulled up on the target. Typically connected to TMS on target CPU.
10	—	NC	This pin is not connected to Flasher ATE.
11	TDI	Output	JTAG data input of target CPU. It is recommended that this pin is pulled to a defined state on the target board. Typically connected to TDI on target CPU.
13	nRES	I/O	Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called “nRST”, “nRESET” or “RESET”.

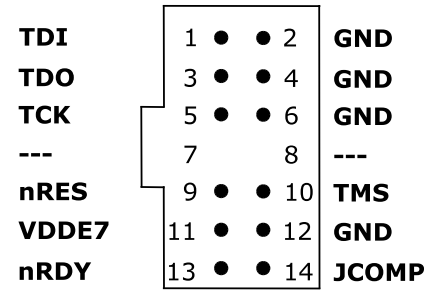
- All pins marked NC are not connected to Flasher ATE. Any signal can be applied here; Flasher ATE will simply ignore such a signal.
- Pins 2, 12, 14 are GND pins connected to GND in Flasher ATE. They should also be connected to GND in the target system.

9.3.3.1 Target power supply

Pin 8 of the 14-pin connector can be used to supply power to the target hardware. Supply voltage is 3.3V / 5V, max. current is 400mA. The output current is monitored and protected against overload and short-circuit. Power can be controlled via the J-Link commander. The ASCII command protocol includes control commands for the power.

9.3.4 Flasher PPC 14-pin adapter

Flasher ATE itself has a 20-pin JTAG connector mounted but comes with a 14-pin adapter for PowerPC devices.



The following table lists the Flasher PPC 14-pin JTAG pinout.

Pin	Signal	Type	Description
1	TDI	Output	JTAG data input of target CPU. It is recommended that this pin is pulled to a defined state on the target board. Typically connected to TDI on target CPU.
3	TDO	Input	JTAG data output from target CPU. Typically connected to TDO on target CPU.
5	TCK	Output	JTAG clock signal to target CPU. It is recommended that this pin is pulled to a defined state on the target board. Typically connected to TCK on target CPU.
7	—	NC	This pin is not connected to Flasher ATE.
8	—	NC	This pin is not connected to Flasher ATE.
9	nRES	I/O	Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called “nRST”, “nRESET” or “RESET”.
10	TMS	Output	JTAG mode set input of target CPU. This pin should be pulled up on the target. Typically connected to TMS on target CPU.
11	VDDE7	Input	This is the target reference voltage. It is used to check if the target has power, to create the logic-level reference for the input comparators and to control the output logic levels to the target. It is normally fed from Vdd of the target board and must not have a series resistor.
13	nRDY	Input	Nexus ready output. Indicates to the development tools that the data is ready to be read from or written to the Nexus read/write access registers.
14	JCOMP	Output	JTAG TAP Controller Enable / JTAG Compliancy (JCOMP). JCOMP is used to enable the TAP controller for communication to the JTAG state machine for boundary scan and for debug access. This pin is set to HIGH by Flasher ATE (in order to enable the JTAG TAP controller on the target device).

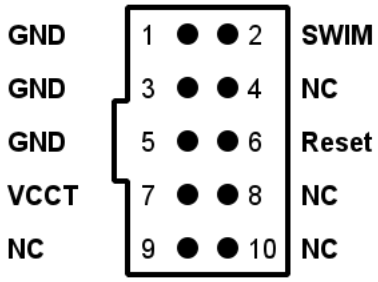
- All pins marked NC are not connected to Flasher ATE. Any signal can be applied here; Flasher ATE will simply ignore such a signal.
- Pins 2, 12, 6, 12 are GND pins connected to GND in Flasher ATE. They should also be connected to GND in the target system.

9.3.5 STM8 adapter

STM8 adapter has a 4-pin and a 10-pin connector to connect a STM8 device to the Flasher ATE.

9.3.5.1 10-pin Connector

The STM8 adapter has a 10-pin connector. The connector is a 10 way Insulation Displacement Connector (IDC) keyed box header (male) that mates with IDC sockets mounted on a ribbon cable.



The signal outputs of the 10-pin interface are the same as for the 4-pin connector. Therefore, if needed for a specific hardware, an adapter cable can be soldered by using a 4-pin “Ernie” cable and soldering one end to a 10-pin header.

10-pin connector pinout

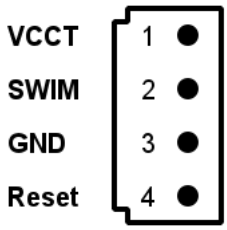
The following table lists the STM8 adapter 10-pin connector pinout:

Pin	Signal	Type	Description
1	GND	I/O	Target power ground line.
2	SWIM	I/O	Open drain output, tristateable, with 100 Ohms series resistor to target. This is the pin used to transfer SWIM input/output data. In case of connection problems the SWIM pin should be pulled up with a 470-680 Ohms series resistor as it can be found in several reference designs.
3	GND	I/O	Target power ground line.
5	GND	I/O	Target power ground line.
6	Reset	I/O	Open collector with 100 Ohms series resistor to target. Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called “nRST”, “nRESET” or “RESET”.
7	VCCT	I/O	Target voltage reference or target supply pin. Has to be connected to the target CPUs supply voltage. It may be used to supply the target board.

Pins 4, 8, 9, 10 are not connected in Flasher STM8. They should also be not connected in the target system.

9.3.5.2 4-pin Connector

Flasher STM8 has a 4-pin connector. The connector is a 4 way connector keyed box header (male) that mates with a connector of type ERNI 214012 mounted on a ribbon cable.



4-pin connector pinout

The following table lists the STM8 adapter 4-pin connector pinout:

Pin	Signal	Type	Description
1	VCCT	I/O	Target voltage reference or target supply pin. Has to be connected to the target CPUs supply voltage. It may be used to supply the target board.
2	SWIM	I/O	Open drain output, tristateable, with 100 Ohms series resistor to target. This is the pin used to transfer SWIM input/output data. In case of connection problems the SWIM pin should be pulled up with a 470-680 Ohms series resistor as it can be found in several reference designs.
3	GND	Output	Target power ground line.
4	Reset	I/O	Open collector with 100 Ohms series resistor to target. Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called “nRST”, “nRESET” or “RESET”.

9.3.5.3 Connection cable

The Flasher ATE is tested with the a 4 way ribbon cable of type ERNI part no. 839016 (100mm) mounted with keyed header (female) mounted on both sides (resulting in a total length of around 120mm). Using your own cable or using a different cable length is not recommended as proper function can not be guaranteed. Using a different cable is on your own risk.

Chapter 10

Support and FAQs

This chapter contains troubleshooting tips together with solutions for common problems which might occur when using theFlasher ATE. There are several steps you can take before contacting support. Performing these steps can solve many problems and often eliminates the need for assistance. This chapter also contains a collection of frequently asked questions (FAQs) with answers.

10.1 Contacting support

Before contacting support, make sure you tried to solve your problem by trying your Flasher ATE with another PC and if possible with another target system to see if it works there. If the device functions correctly, the USB setup on the original machine or your target hardware is the source of the problem, not Flasher.

If you need to contact support, send the following information to *ticket_flasher@segger.com*

- A detailed description of the problem
- Flasher ATE serial number
- Information about your target hardware (processor, board, etc.).
- `FLASHER.JFLASH`, `FLASHER.CFG`, `FLASHER.DAT` (if possible), `FLASHER.LOG`, `SERIAL.TXT` file from Flasher. To get these files, please download them via FTP.

Flasher ATE is sold directly by SEGGER.

10.2 Frequently Asked Questions

Maximum JTAG speed

Q: What is the maximum JTAG speed supported by Flasher?
A: Flasher's maximum supported JTAG speed is 15MHz.

Maximum download speed

Q: What is the maximum download speed?
A: The maximum download speed is currently about 720 Kbytes/second when downloading into RAM. The actual speed depends on various factors, such as JTAG, clock speed, host CPU core etc.

IP address setup

Q: My Flasher ATE is not listed in the J-Link Configurator.
A: If your computer is connected to a wireless LAN, some routers do not forward the broadcast messages used to find the Flasher ATE from LAN to WLAN. In that case either use a LAN connection for your computer or connect the Flasher ATE via USB to your computer and choose the Flasher ATE in the list of connected USB devices.

Adapter

Q: Which adapter do I need?
A: There is a list in our wiki, which lists all cores and adapter. For details please take a closer look here: *https://wiki.segger.com/Flasher_ATE*.

Chapter 11

Mechanics

If you need to mount the Flasher ATE into the production environment, a drawing of the mechanical layout can downloaded here:
https://www.segger.com/downloads/flasher/Flasher_ATE_MountingHoles.

Chapter 12

Glossary

This chapter describes important terms used throughout this manual.

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.

Cache cleaning

The process of writing dirty data in a cache to main memory.

Coprocessor

An additional processor that is used for certain operations, for example, for floating-point math calculations, signal processing, or memory management.

Dirty data

When referring to a processor data cache, data that has been written to the cache but has not been written to main memory is referred to as dirty data. Only write-back caches can have dirty data because a write-through cache writes data to the cache and to main memory simultaneously. See also cache cleaning.

Halfword

A 16-bit unit of information.

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.

ID

Identifier.

IEEE 1149.1

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

Image

An executable file that has been loaded onto a processor for execution.

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.

Memory management unit (MMU)

Hardware that controls caches and access permissions to blocks of memory, and translates virtual to physical addresses.

Memory Protection Unit (MPU)

Hardware that controls access permissions to blocks of memory. Unlike an MMU, a MPU does not translate virtual addresses to physical addresses.

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.

Remapping

Changing the address of physical memory or devices after the application has started executing. This is typically done to make RAM replace ROM once the initialization has been done.

RTOS

Real Time Operating System.

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 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 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.

Word

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

Chapter 13

Literature and references

This chapter lists documents, which we think may be useful to gain a deeper under- standing of technical details.

Reference	Title	Comments
[J-Link]	J-Link / J-Trace User Guide	This document describes J-Link and J-Trace. It is publicly available from SEGGER (*https://www.segger.com*).
[J-Flash]	J-Flash User Guide	This document describes J-Flash. It is publicly available from SEGGER (*https://www.segger.com*).
[Flasher ATE]	Flasher ATE Getting Started	Step by step guide to get the first project running with the Flasher ATE (*https://www.segger.com/flasher-ate*).
[Flasher ATE wiki]	Flasher ATE wiki pages	*https://wiki.segger.com/Flasher_ATE*.

