

Flasher User Guide



Software Version V6.00
Manual Rev. 0

Date: September 22, 2016

Document: UM08022



A product of SEGGER Microcontroller GmbH & Co. KG

www.segger.com

Disclaimer

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

Copyright notice

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

© 2016 SEGGER Microcontroller GmbH & Co. KG, Hilden / Germany

Trademarks

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

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

Contact address

SEGGER Microcontroller GmbH & Co. KG

In den Weiden 11

D-40721 Hilden

Germany

Tel. +49 2103-2878-0

Fax. +49 2103-2878-28

Email: support@segger.com

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

Revisions

This manual describes the Flasher device.

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

Revision	Date	By	Explanation
V6.10a Rev. 0	160922	EL	Chapter "Introduction" * Section "Flasher Portable" updated.
V6.00 Rev. 0	160715	NG	Chapter "Working with Flasher" * Section "Batch Programming" added.
V5.12e Rev. 0	160511	NG	Chapter "Working with Flasher" * Section "Setting up Flasher for stand-alone mode" moved. * Section "Preparing for stand-alone operation manually" added.
V5.02f Rev. 0	151023	RH	Chapter "Remote control" * Section "General usage" added.
V5.02e Rev. 0	151021	EL	Chapter "Introduction" * Section "Specifications" updated for all models.

Revision	Date	By	Explanation
V5.02f Rev. 0	151014	RH	Chapter "Working with Flasher" * Section "Programming multiple targets" added.
V4.50c Rev. 0	150611	EL	Chapter "Working with Flasher" * Section "Programming multiple targets in parallel" added.
V4.98 Rev. 0	150205	AG	Chapter "Working with Flasher" * Section "Authorized flashing" added. * Section "Limiting the number of programming cycles" added. * Section "Operating Modes" updated
V4.86 Rev. 0	140610	AG	Chapter "Working with Flasher" * Section "Newline encoding" added.
V4.80 Rev. 1	131220	AG	Chapter "Working with Flasher" * Section "Multiple File Support" updated.
V4.80 Rev. 0	131031	EL	Chapter "Remote control" * Section "Commands to Flasher" updated. #FCRC command added.
V4.78 Rev. 0	130917	AG	Chapter "Introduction" * Section "Features of Flasher Portable" added. Chapter "Working with Flasher" * Section "Flasher Portable" added. * Section "Multiple File Support" updated.
V4.72 Rev. 0	130612	EL	Chapter "Working with Flasher" * Section "Patch file support" added.
V4.64a Rev. 0	130226	EL	Chapter "Working with Flasher" * Section "LED status indicators" updated.
V4.63a Rev. 0	130131	EL	Chapter "Remote Control" * Section "ASCII command interface" Chapter "ASCII interface via Telnet" added.
V4.62 Rev. 0	130125	EL	Flasher ARM, Flasher RX and Flasher PPC manual have been combined.
V5.02 Rev 0	150807	RE	New commands #FLIST and #MKDIR in ASCII command interface

About this document

This document describes the Flasher family (Flasher ARM, Flasher RX and Flasher PPC). It provides an overview about the major features of the Flasher, gives some background information about JTAG and describes Flasher related software packages available from Segger. Finally, the chapter *Support and FAQs* on page 85 helps to troubleshoot common problems.

For simplicity, we will refer to Flasher ARM/RX/PPC as Flasher in this manual.

Typographic conventions

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).
Reference	Reference to chapters, tables and figures or other documents.
GUIElement	Buttons, dialog boxes, menu names, menu commands.

Table 1.1: Typographic conventions



SEGGER Microcontroller GmbH & Co. KG develops and distributes software development tools and ANSI C software components (middleware) for embedded systems in several industries such as telecom, medical technology, consumer electronics, automotive industry and industrial automation.

SEGGER's intention is to cut software development-time for embedded applications by offering compact flexible and easy to use middleware, allowing developers to concentrate on their application.

Our most popular products are emWin, a universal graphic software package for embedded applications, and embOS, a small yet efficient real-time kernel. emWin, written entirely in ANSI C, can easily be used on any CPU and most any display. It is complemented by the available PC tools: Bitmap Converter, Font Converter, Simulator and Viewer. embOS supports most 8/16/32-bit CPUs. Its small memory footprint makes it suitable for single-chip applications.

Apart from its main focus on software tools, SEGGER develops and produces programming tools for flash microcontrollers, as well as J-Link, a JTAG emulator to assist in development, debugging and production, which has rapidly become the industry standard for debug access to ARM cores.

Corporate Office:

<http://www.segger.com>

United States Office:

<http://www.segger-us.com>

EMBEDDED SOFTWARE (Middleware)



emWin

Graphics software and GUI

emWin is designed to provide an efficient, processor- and display controller-independent graphical user interface (GUI) for any application that operates with a graphical display. Starterkits, eval- and trial-versions are available.



embOS

Real Time Operating System

embOS is an RTOS designed to offer the benefits of a complete multitasking system for hard real time applications with minimal resources. The profiling PC tool embOSView is included.



emFile

File system

emFile is an embedded file system with FAT12, FAT16 and FAT32 support. emFile has been optimized for minimum memory consumption in RAM and ROM while maintaining high speed. Various Device drivers, e.g. for NAND and NOR flashes, SD/MMC and CompactFlash cards, are available.



emUSB

USB device stack

A USB stack designed to work on any embedded system with a USB client controller. Bulk communication and most standard device classes are supported.

SEGGER TOOLS

Flasher

Flash programmer

Flash Programming tool primarily for microcontrollers.

J-Link

JTAG emulator for ARM cores

USB driven JTAG interface for ARM cores.

J-Trace

JTAG emulator with trace

USB driven JTAG interface for ARM cores with Trace memory. supporting the ARM ETM (Embedded Trace Macrocell).

J-Link / J-Trace Related Software

Add-on software to be used with SEGGER's industry standard JTAG emulator, this includes flash programming software and flash breakpoints.



Table of Contents

1	Introduction	9
1.1	Flasher overview	10
1.1.1	Features of Flasher ARM/PPC/RX	10
1.1.2	Features of Flasher Portable	11
1.1.3	Working environment	11
1.2	Specifications	12
1.2.1	Specifications for Flasher ARM	12
1.2.2	Specifications for Flasher RX	14
1.2.3	Specifications for Flasher PPC	16
1.2.4	Specifications for Flasher Portable	18
2	Working with Flasher	21
2.1	Flasher Portable	22
2.1.1	Housing & Buttons	23
2.2	Setting up the IP interface	25
2.2.1	Connecting the first time	25
2.3	Operating modes	26
2.3.1	J-Link mode	26
2.3.2	Stand-alone mode	30
2.3.3	MSD mode	32
2.4	Setting up Flasher for stand-alone mode	33
2.4.1	Preparing for stand-alone operation manually	36
2.5	Multiple File Support	38
2.5.1	Flasher Portable specifics	38
2.5.2	Example	39
2.6	Programming multiple targets	40
2.6.1	Programming multiple targets with J-Flash	40
2.7	Batch Programming in stand-alone mode	41
2.7.1	Flasher Portable specifics	41
2.7.2	Examples	42
2.8	Serial number programming	43
2.8.1	Serial number settings	43
2.8.2	Serial number file	44
2.8.3	Serial number list file	44
2.8.4	Programming process	45
2.8.5	Downloading serial number files to Flasher	46
2.8.6	Sample setup	46
2.9	Patch file support	48
2.10	Limiting the number of programming cycles	49
2.10.1	Changed fail/error LED indicator behavior	49
2.10.2	Required Flasher hardware version for Cntdown.txt support	49
2.11	Authorized flashing	50
2.11.1	Creating / Adding the secure area	50
2.11.2	Moving files to the secure area	51
2.11.3	Considerations to be taken when using the secure area	52
2.11.4	Required Flasher hardware version	52
2.12	Target interfaces	53
2.13	Supported microcontrollers	54
2.13.1	Flasher	54
2.14	Support of external flashes	55
2.14.1	Flasher ARM	55

2.14.2	Flasher RX.....	55
2.14.3	Flasher PPC	55
2.15	Supported cores.....	56
2.15.1	Flasher ARM	56
2.15.2	Flasher RX.....	56
2.15.3	Flasher PPC	56
2.16	Newline encoding	57
2.17	Programming multiple targets in parallel	58
3	Remote control.....	59
3.1	Overview	60
3.2	Handshake control	61
3.3	ASCII command interface.....	62
3.3.1	Introduction	62
3.3.2	General command and reply message format.....	62
3.3.3	General usage	62
3.3.4	Settings for ASCII interface via RS232	62
3.3.5	Settings for ASCII interface via Telnet.....	62
3.3.6	Commands to Flasher	63
3.3.7	Reply from Flasher	68
4	Performance	71
4.1	Performance of MCUs with internal flash memory	72
4.1.1	Flasher ARM	72
4.1.2	Flasher RX.....	72
4.1.3	Flasher PPC	72
5	Hardware	73
5.1	Flasher ARM 20-pin JTAG/SWD Connector	74
5.1.1	Pinout JTAG	74
5.1.2	Pinout SWD	75
5.1.3	Target power supply	75
5.2	Flasher RX 14-pin connector	77
5.2.1	Target power supply	77
5.3	Flasher PPC 14-pin connector.....	79
5.4	Target board design	80
5.4.1	Pull-up/pull-down resistors	80
5.4.2	RESET, nTRST	80
5.5	Adapters	81
5.5.1	JTAG Isolator.....	81
5.5.2	J-Link Needle Adapter.....	82
5.6	How to determine the hardware version	83
6	Support and FAQs	85
6.1	Contacting support.....	86
6.2	Frequently Asked Questions.....	87
7	Background information	89
7.1	Flash programming	90
7.1.1	How does flash programming via Flasher work ?.....	90
7.1.2	Data download to RAM.....	90
7.1.3	Available options for flash programming	90
8	Glossary.....	91
9	Literature and references.....	95

Chapter 1

Introduction

This chapter gives a short overview about the different models of the Flasher family and their features.

1.1 Flasher overview

Flasher is a programming tool for microcontrollers with on-chip or external flash memory. Flasher is designed for programming flash targets with the J-Flash software or stand-alone. In addition to that Flasher can also be used as a regular J-Link. For more information about J-Link in general, please refer to the *J-Link / J-Trace User Guide* which can be downloaded at <http://www.segger.com>.

Flasher connects to a PC using the USB/Ethernet/RS232 interface (what host interfaces are available depends on the Flasher model), running Microsoft Windows 2000, Windows XP, Windows 2003, Windows Vista, Windows 7 or Windows 8. In stand-alone mode, Flasher can be driven by the start/stop button, or via the RS232 interface (handshake control or ASCII interface). Flasher always has a 20-pin connector, which target interfaces are supported depends on the Flasher model:

- For Flasher ARM: JTAG and SWD are supported.
- For Flasher RX: JTAG is supported. Flasher comes with additional 14-pin RX adapter
- For Flasher PPC: JTAG is supported. Flasher comes with additional 14-pin PPC adapter.

1.1.1 Features of Flasher ARM/PPC/RX

- Three boot modes: J-Link mode, stand-alone mode, MSD mode
- Stand-alone JTAG/SWD programmer (Once set up, Flasher can be controlled without the use of PC program)
- No power supply required, powered through USB
- Supports internal and external flash devices
- 64 MB memory for storage of target program
- Can be used as J-Link (emulator) with a download speed of up to 720 Kbytes/second
- Serial in target programming supported
- Data files can updated via USB/Ethernet (using the J-Flash software), via RS232 or via the MSD functionality of Flasher

Flasher model	Supported cores	Supported target interfaces	Flash programming speed (depending on target hardware)
Flasher ARM	ARM7/ARM9/Cortex-M	JTAG, SWD	between 30-300 Kbytes/second
Flasher RX	Renesas RX610, RX621, RX62N, RX62T	JTAG	between 170 and 300 Kbytes/second
Flasher PPC	Power PC e200z0	JTAG	up to 138 Kbytes/second

1.1.2 Features of Flasher Portable

- Stand-alone in-circuit-programmer (Once set up, Flasher can be controlled without the use of a PC program)
- Powered by standard batteries, no Laptop or external power supply required.
- Multiple firmware images can be stored on Flasher
- 128 MB memory for storage of target program
- Easy selection of image to be programmed, via button
- Supported CPUs: ARM Cortex, Legacy ARM7/9, Renesas RX, Freescale PowerPC
- Supports internal and external flash
- Free software updates*, 1 year of support
- Data files can be updated as mass storage or via J-Flash
- Programming speed between 30-300 Kbytes/second (actual speed depends on target hardware)

Note: Ethernet and RS232 as host interface are not available for Flasher Portable

Note: *As a legitimate owner of a SEGGER Flasher, you can always download the latest software free of charge. Though not planned and not likely, we reserve the right to change this policy. Note that older models may not be supported by newer versions of the software. Typically, we support older models with new software at least 3 years after end of life.

Supported cores	Supported target interfaces	Flash programming speed (depending on target hardware)
ARM7/ARM9/Cortex-M	JTAG, SWD	between 30-300 Kbytes/second
Renesas RX610, RX621, RX62N, RX62T	JTAG	between 170 and 300 Kbytes/second
Power PC e200z0	JTAG	up to 138 Kbytes/second

1.1.3 Working environment

General

Flasher can operate 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 2000, Windows XP, Windows 2003, Windows Vista, Windows 7 or Windows 8. It needs to have a USB, Ethernet or RS232 interface available for communication with Flasher.

Power supply

Flasher Portable: 3x standard AAA batteries or 5V DC, min. 100 mA via USB connector.

Other Flashers: 5V DC, min. 100 mA via USB connector.

Installing 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 always be downloaded from our website: http://www.segger.com/download_jlink.html. For more information about using J-Flash please refer to *UM08003_JFlashARM.pdf* (J-Flash user guide) which is also available for download on our website.

1.2 Specifications

1.2.1 Specifications for Flasher ARM

General	
Supported OS	Microsoft Windows 2000 Microsoft Windows XP Microsoft Windows XP x64 Microsoft Windows 2003 Microsoft Windows 2003 x64 Microsoft Windows Vista Microsoft Windows Vista x64 Microsoft Windows 7 Microsoft Windows 7 x64 Microsoft Windows 8 Microsoft Windows 8 x64
Operating Temperature	+5 °C ... +60 °C
Storage Temperature	-20 °C ... +60 °C
Relative Humidity (non-condensing)	<90% rH
Mechanical	
Size (without cables)	121mm x 66mm x 30mm
Weight (without cables)	119g
Available interfaces	
USB Host interface	USB 2.0, full speed
Ethernet Host interface	10/100 MBit
RS232 Host interface	RS232 9-pin
Target interface	JTAG 20-pin (14-pin adapter available)
JTAG Interface, Electrical	
Power Supply	USB powered, 100mA for Flasher ARM. 500 mA if target is powered by Flasher ARM
Target interface voltage (V _{IF})	1.2 ... 5V
Target supply voltage	Supply voltage is 5V, max.
Target supply current	max. 400mA
Reset Type	Open drain. Can be pulled low or tristated
Reset low level output voltage (V _{OL})	V _{OL} ≤ 10% of V _{IF}
For the whole target voltage range (1.8V ≤ V _{IF} ≤ 5V)	
LOW level input voltage (V _{IL})	V _{IL} ≤ 40% of V _{IF}
HIGH level input voltage (V _{IH})	V _{IH} ≥ 60% of V _{IF}
For 1.8V ≤ V _{IF} ≤ 3.6V	
LOW level output voltage (V _{OL}) with a load of 10 kOhm	V _{OL} ≤ 10% of V _{IF}
HIGH level output voltage (V _{OH}) with a load of 10 kOhm	V _{OH} ≥ 90% of V _{IF}
For 3.6 ≤ V _{IF} ≤ 5V	
LOW level output voltage (V _{OL}) with a load of 10 kOhm	V _{OL} ≤ 20% of V _{IF}
HIGH level output voltage (V _{OH}) with a load of 10 kOhm	V _{OH} ≥ 80% of V _{IF}

Table 1.1: Flasher ARM specifications

JTAG Interface, Timing	
Max. JTAG speed	up to 12MHz
Data input rise time (T_{rdi})	$T_{rdi} \leq 20\text{ns}$
Data input fall time (T_{fdi})	$T_{fdi} \leq 20\text{ns}$
Data output rise time (T_{rdo})	$T_{rdo} \leq 10\text{ns}$
Data output fall time (T_{fdo})	$T_{fdo} \leq 10\text{ns}$
Clock rise time (T_{rc})	$T_{rc} \leq 10\text{ns}$
Clock fall time (T_{fc})	$T_{fc} \leq 10\text{ns}$

Table 1.1: Flasher ARM specifications

1.2.1.1 Flasher ARM download speed

The following table lists Flasher ARM performance values (Kbytes/second) for writing to memory (RAM) via the JTAG interface:

Hardware	ARM7 memory download
Flasher ARM	720 Kbytes/s (12MHz JTAG)

Table 1.2: Download speed of Flasher ARM

Note: The actual speed depends on various factors, such as JTAG, clock speed, host CPU core etc.

1.2.2 Specifications for Flasher RX

General	
Supported OS	Microsoft Windows 2000 Microsoft Windows XP Microsoft Windows XP x64 Microsoft Windows 2003 Microsoft Windows 2003 x64 Microsoft Windows Vista Microsoft Windows Vista x64 Microsoft Windows 7 Microsoft Windows 7 x64 Microsoft Windows 8 Microsoft Windows 8 x64
Operating Temperature	+5 °C ... +60 °C
Storage Temperature	-20 °C ... +60 °C
Relative Humidity (non-condensing)	<90% rH
Mechanical	
Size (without cables)	121mm x 66mm x 30mm
Weight (without cables)	119g
Available interfaces	
USB Host interface	USB 2.0, full speed
Ethernet Host interface	10/100 MBit
RS232 Host interface	RS232 9-pin
Target interface	JTAG 20-pin (shipped with 14-pin adapter for Renesas RX)
JTAG Interface, Electrical	
Power Supply	USB powered, 100mA for Flasher ARM. 500 mA if target is powered by Flasher ARM
Target interface voltage (V _{IF})	1.2 ... 5V
Target supply voltage	Supply voltage is 5V, max. (on the J-Link RX 14-pin adapter, the target supply voltage can be switched between 3.3V and 5V)
Target supply current	max. 400mA
Reset Type	Open drain. Can be pulled low or tristated
Reset low level output voltage (V _{OL})	V _{OL} ≤ 10% of V _{IF}
For the whole target voltage range (1.8V ≤ V _{IF} ≤ 5V)	
LOW level input voltage (V _{IL})	V _{IL} ≤ 40% of V _{IF}
HIGH level input voltage (V _{IH})	V _{IH} ≥ 60% of V _{IF}
For 1.8V ≤ V _{IF} ≤ 3.6V	
LOW level output voltage (V _{OL}) with a load of 10 kOhm	V _{OL} ≤ 10% of V _{IF}
HIGH level output voltage (V _{OH}) with a load of 10 kOhm	V _{OH} ≥ 90% of V _{IF}
For 3.6 ≤ V _{IF} ≤ 5V	
LOW level output voltage (V _{OL}) with a load of 10 kOhm	V _{OL} ≤ 20% of V _{IF}

Table 1.3: Flasher RX specifications

HIGH level output voltage (V_{OH}) with a load of 10 kOhm	$V_{OH} \geq 80\%$ of V_{IF}
JTAG Interface, Timing	
Max. JTAG speed	up to 12MHz
Data input rise time (T_{rdi})	$T_{rdi} \leq 20\text{ns}$
Data input fall time (T_{fdi})	$T_{fdi} \leq 20\text{ns}$
Data output rise time (T_{rdo})	$T_{rdo} \leq 10\text{ns}$
Data output fall time (T_{fdo})	$T_{fdo} \leq 10\text{ns}$
Clock rise time (T_{rc})	$T_{rc} \leq 10\text{ns}$
Clock fall time (T_{fc})	$T_{fc} \leq 10\text{ns}$

Table 1.3: Flasher RX specifications

1.2.2.1 Flasher RX download speed

The following table lists Flasher RX performance values (Kbytes/second) for writing to memory (RAM) via the JTAG interface:

Hardware	Flasher RX600 series memory download
Flasher RX	720 Kbytes/s (12MHz JTAG)

Table 1.4: Download speed of Flasher RX

Note: The actual speed depends on various factors, such as JTAG, clock speed, host CPU core etc.

1.2.3 Specifications for Flasher PPC

General	
Supported OS	Microsoft Windows 2000 Microsoft Windows XP Microsoft Windows XP x64 Microsoft Windows 2003 Microsoft Windows 2003 x64 Microsoft Windows Vista Microsoft Windows Vista x64 Microsoft Windows 7 Microsoft Windows 7 x64 Microsoft Windows 8 Microsoft Windows 8 x64
Operating Temperature	+5 °C ... +60 °C
Storage Temperature	-20 °C ... +60 °C
Relative Humidity (non-condensing)	<90% rH
Mechanical	
Size (without cables)	121mm x 66mm x 30mm
Weight (without cables)	119g
Available interfaces	
USB Host interface	USB 2.0, full speed
Ethernet Host interface	10/100 MBit
RS232 Host interface	RS232 9-pin
Target interface	JTAG 20-pin (shipped with 14-pin adapter for Renesas PPC)
JTAG Interface, Electrical	
Power Supply	USB powered, 100mA for Flasher ARM. 500 mA if target is powered by Flasher ARM
Target interface voltage (V _{IF})	1.2 ... 5V
Target supply voltage	Supply voltage is 5V, max.
Target supply current	max. 400mA
Reset Type	Open drain. Can be pulled low or tristated
Reset low level output voltage (V _{OL})	V _{OL} ≤ 10% of V _{IF}
For the whole target voltage range (1.8V ≤ V _{IF} ≤ 5V)	
LOW level input voltage (V _{IL})	V _{IL} ≤ 40% of V _{IF}
HIGH level input voltage (V _{IH})	V _{IH} ≥ 60% of V _{IF}
For 1.8V ≤ V _{IF} ≤ 3.6V	
LOW level output voltage (V _{OL}) with a load of 10 kOhm	V _{OL} ≤ 10% of V _{IF}
HIGH level output voltage (V _{OH}) with a load of 10 kOhm	V _{OH} ≥ 90% of V _{IF}
For 3.6 ≤ V _{IF} ≤ 5V	
LOW level output voltage (V _{OL}) with a load of 10 kOhm	V _{OL} ≤ 20% of V _{IF}
HIGH level output voltage (V _{OH}) with a load of 10 kOhm	V _{OH} ≥ 80% of V _{IF}
JTAG Interface, Timing	

Table 1.5: Flasher PPC specifications

Max. JTAG speed	up to 12MHz
Data input rise time (T_{rdi})	$T_{rdi} \leq 20\text{ns}$
Data input fall time (T_{fdi})	$T_{fdi} \leq 20\text{ns}$
Data output rise time (T_{rdo})	$T_{rdo} \leq 10\text{ns}$
Data output fall time (T_{fdo})	$T_{fdo} \leq 10\text{ns}$
Clock rise time (T_{rc})	$T_{rc} \leq 10\text{ns}$
Clock fall time (T_{fc})	$T_{fc} \leq 10\text{ns}$

Table 1.5: Flasher PPC specifications

1.2.3.1 Flasher PPC download speed

The following table lists Flasher PPC performance values (Kbytes/second) for writing to memory (RAM) via the JTAG interface:

Hardware	Memory download
Flasher PPC	530 Kbytes/s (8 MHz JTAG)

Table 1.6: Download speed of Flasher PPC

Note: The actual speed depends on various factors, such as JTAG, clock speed, host CPU core etc.

1.2.4 Specifications for Flasher Portable

General	
Supported OS	Microsoft Windows 2000 Microsoft Windows XP Microsoft Windows XP x64 Microsoft Windows 2003 Microsoft Windows 2003 x64 Microsoft Windows Vista Microsoft Windows Vista x64 Microsoft Windows 7 Microsoft Windows 7 x64 Microsoft Windows 8 Microsoft Windows 8 x64
Operating Temperature	+5 °C ... +60 °C
Storage Temperature	-20 °C ... +65 °C
Relative Humidity (non-condensing)	<90% rH
Power Supply	a) USB powered, 100mA for Flasher ARM. 500 mA if target is powered by Flasher ARM b) Batteries powered (3xAAA)
Mechanical	
Size (without cables)	130mm x 65mm x 25mm
Weight (without cables)	120g
Available interfaces	
USB Host interface	USB 2.0
Target interface	Standard 20-pin 0.1" connector (Adapters available).
Target Interface, Electrical	
Target interface voltage (V _{IF})	1.2 ... 5V
Target supply voltage	Supply voltage is 4.5V, max. (depends on the current battery voltage).
Target supply current	max. 400mA
Reset Type	Open drain. Can be pulled low or tristated
Reset low level output voltage (V _{OL})	V _{OL} ≤ 10% of V _{IF}
For the whole target voltage range (1.8V ≤ V _{IF} ≤ 5V)	
LOW level input voltage (V _{IL})	V _{IL} ≤ 40% of V _{IF}
HIGH level input voltage (V _{IH})	V _{IH} ≥ 60% of V _{IF}
For 1.8V ≤ V _{IF} ≤ 3.6V	
LOW level output voltage (V _{OL}) with a load of 10 kOhm	V _{OL} ≤ 10% of V _{IF}
HIGH level output voltage (V _{OH}) with a load of 10 kOhm	V _{OH} ≥ 90% of V _{IF}
For 3.6 ≤ V _{IF} ≤ 5V	
LOW level output voltage (V _{OL}) with a load of 10 kOhm	V _{OL} ≤ 20% of V _{IF}
HIGH level output voltage (V _{OH}) with a load of 10 kOhm	V _{OH} ≥ 80% of V _{IF}
JTAG Interface, Timing	

Table 1.7:

Max. JTAG speed	up to 12MHz
Data input rise time (T_{rdi})	$T_{rdi} \leq 20\text{ns}$
Data input fall time (T_{fdi})	$T_{fdi} \leq 20\text{ns}$
Data output rise time (T_{rdo})	$T_{rdo} \leq 10\text{ns}$
Data output fall time (T_{fdo})	$T_{fdo} \leq 10\text{ns}$
Clock rise time (T_{rc})	$T_{rc} \leq 10\text{ns}$
Clock fall time (T_{fc})	$T_{fc} \leq 10\text{ns}$

Table 1.7:

1.2.4.1 Supported CPU cores

1.2.4.2 Supported CPU cores

Flasher Portable supports the following CPU cores:

ARM Cortex

- Cortex-A5
- Cortex-A8
- Cortex-A9
- Cortex-R4
- Cortex-R5
- Cortex-M0
- Cortex-M0+
- Cortex-M1
- Cortex-M3
- Cortex-M4

ARM (legacy cores)

- ARM720T
- ARM7TDMI
- ARM7TDMI-S
- ARM920T
- ARM922T
- ARM926EJ-S
- ARM946E-S
- ARM966E-S
- ARM1136JF-S
- ARM1136J-S
- ARM1156T2-S
- ARM1156T2F-S
- ARM1176JZ-S
- ARM1176JZF
- ARM1176JZF-S

Renesas RX

- RX111
- RX210
- RX220
- RX21A
- RX610
- RX621
- RX62G
- RX62N
- RX62T
- RX630
- RX631
- RX63N
- RX63T

Freescal Power PC

- e200z0

1.2.4.3 Supported Target interfaces

Flasher Portable supports the following target interfaces:

- JTAG
- SWD
- FINE
- SPD

Chapter 2

Working with Flasher

This chapter describes functionality and how to use Flasher.

2.1 Flasher Portable

Flasher Portable is a portable version of SEGGERs Flasher family, which has been designed to fill the need of an extremely portable, production grade, Flash programmer used for in-field firmware updates. No need to be tethered to an outlet, it is powered by three standard AAA batteries. Flasher Portable programs flash targets in stand-alone mode or via J-Flash PC software.

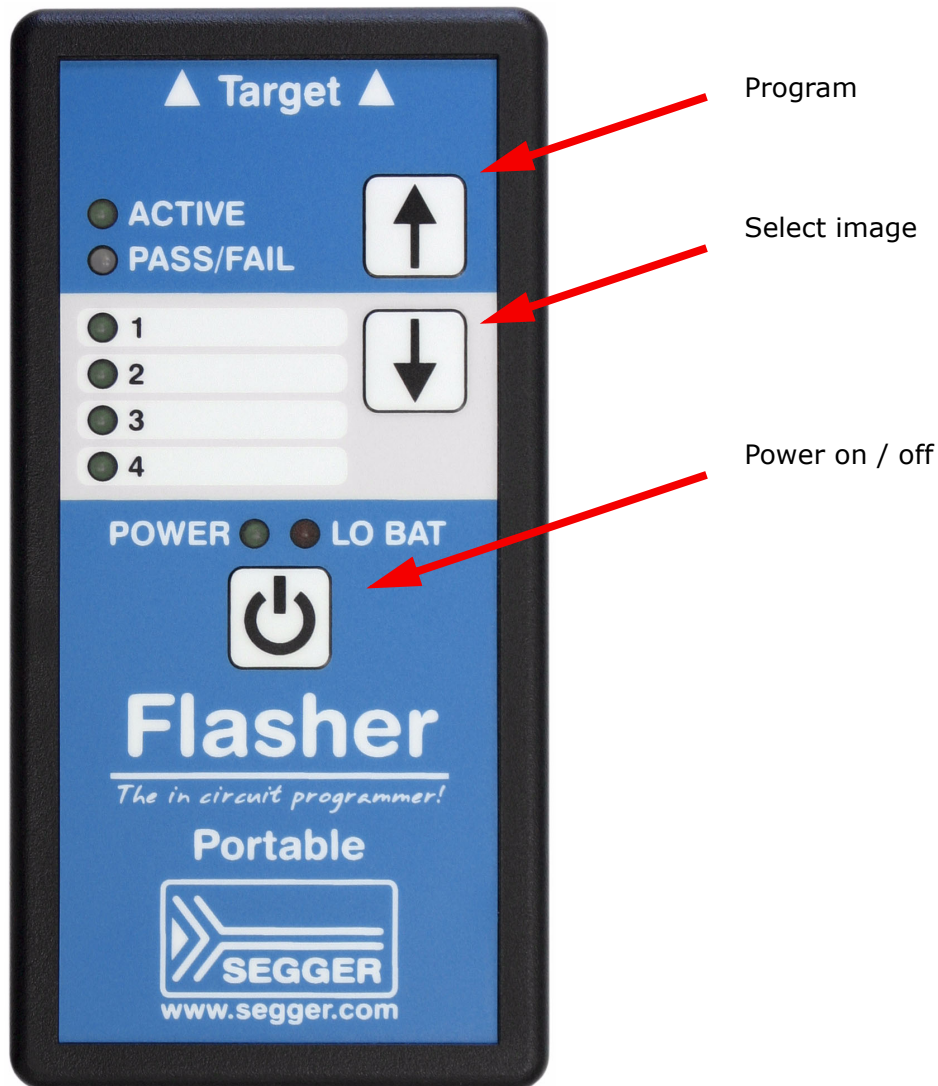
Furthermore Flasher Portable allows the user to select between four data images to be programmed. The images can be easily selected by using the arrow buttons on the front of the housing. For more information about support for multiple images, please refer to *Multiple File Support* on page 38.

For setup and configuration purposes, the Flasher Portable connects to a PC via USB interface, running Microsoft Windows 2000, Windows XP, Windows 2003, Windows Vista, Windows 7 or Windows 8 and has a built-in standard 20-pin J-Link target connector.

Note: Ethernet and RS232 as host interface are not available for Flasher Portable.

2.1.1 Housing & Buttons

Flasher Portable comes with several leds which are for example used to represent the status of an ongoing flash progress. Furthermore, there are three buttons which allow the user to control Flasher Portable. For a detailed description of the functions, take a look at the two tables below :



LED	Description
ACTIVE	GREEN. Blinks while Flasher Portable is busy / performs operations on the target.
PASS/FAIL	GREEN/RED. Indicates, if last flashing cycle was successful. <ul style="list-style-type: none"> GREEN: Flashing cycle completed successfully. RED: Flashing cycle completed with error.
1/2/3/4	GREEN: Indicates which image is currently selected for programming. For more information about multiple image support, please refer to <i>Multiple File Support</i> on page 38.
POWER	GREEN: Indicates, if Flasher is currently powered. Blinks while Flasher Portable tries to enumerate via USB.
LO BAT	1) The LED is off, meaning battery voltage is > 3.3V 2) The red LED is constant on if battery voltage is low (<= 3.3V). In this mode, Flasher still allows programming. 3) The red LED starts blinking in case of the battery voltage is below 3.0V. Flasher refuse programming attempts.

Button	Description
Program	Start programming process with currently selected image.
Select image	Select image to be programmed next time the <code>Program</code> button it pressed.
Power on/off	Used to power-on / power-off Flasher Portable. Please note that to power up Flasher Portable, the button should be hold for at least 1 second to make sure software can boot and take control of power circuit, so Flasher Portable keeps powered, after releasing the button.

2.2 Setting up the IP interface

Some of the Flasher models come with an additional Ethernet interface to communicate with the host system. These Flashers also come with 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 the first time

When connecting Flasher 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 is, to use the J-Link Configurator. The J-Link Configurator is a small GUI-based utility which shows a list of all emulator 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 setup of the IP interface of Flasher is the same as for other emulators of the J-Link family. For more information about how to setup the IP interface of Flasher, please refer to *UM08001, J-Link / J-Trace User Guide*, chapter *Setup*, section *Setting up the IP interface*. For more information about how to use Flasher via Ethernet or prepare Flasher via Ethernet for stand-alone mode, please refer to *Operating modes* on page 26.

2.3 Operating modes

Flasher is able to boot in 3 different modes:

- J-Link mode
- Stand-alone mode
- MSD (Mass storage device) mode

Definition J-Link mode

Flasher is connected to a PC via USB/Ethernet and controlled by a PC application (J-Flash). If there is an RS232 connection to a PC, does not have any influence on if J-Link mode is entered or not. In this mode, Flasher can be used as a J-Link and controlled by the software in the J-Link software and documentation package (J-Link Commander, J-Flash, ...)

Definition Stand-alone mode

This mode is entered when there is no active USB/Ethernet connection to a host PC, e.g. if Flasher is only powered via a USB power supply.

Definition MSD mode

Entered only if Flasher Start/Stop button (on Flasher Portable the "PROG" button) is kept pressed for at least 2 seconds while connection Flasher via USB. In this mode, Flasher enumerates as a mass storage device (like an USB Stick) at the host PC. In this mode, configuration + data files can be manually placed on the Flasher and the flasher logfile can be read out.

2.3.1 J-Link mode

If you want to use Flasher for the first time you need to install the J-Link software and documentation package. After installation, connect Flasher to the host PC via USB or Ethernet. For more information about how to install the J-Link software and documentation package please refer to the *J-Link / J-Trace User Guide, chapter Setup* which can be downloaded from http://www.segger.com/download_jlink.html.

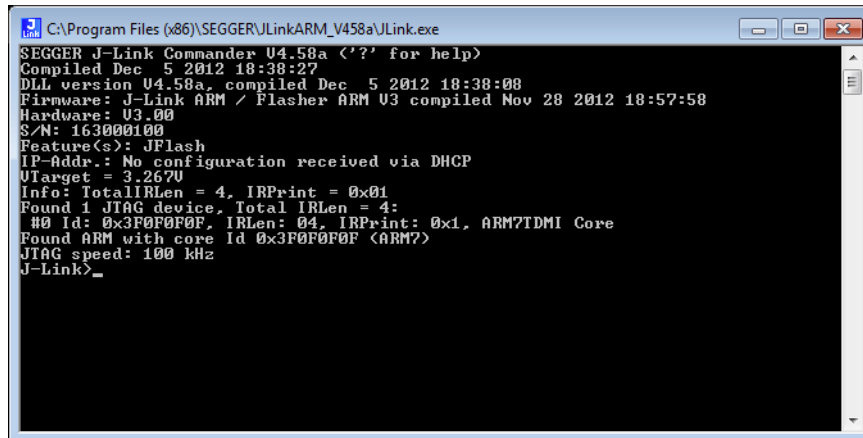
2.3.1.1 Connecting the target system

Power-on sequence

In general, Flasher should be powered on before connecting it with the target device. That means you should first connect Flasher with the host system via USB / Ethernet and then connect Flasher with the target device via JTAG or SWD. Power-on the device after you connected Flasher to it. Flasher will boot in "J-Link mode".

Verifying target device connection with J-Link.exe

If the USB driver is working properly and your Flasher is connected with the host system, you may connect Flasher to your target hardware. Then start the J-Link command line tool `JLink.exe`, which should now display the normal Flasher related information and in addition to that it should report that it found a JTAG target and the targets core ID. The screenshot below shows the output of `JLink.exe`.



```

C:\Program Files (x86)\SEGGER\JLinkARM_V458a\JLink.exe
SEGGER J-Link Commander V4.58a '?' for help>
Compiled Dec 5 2012 18:38:27
DLL version V4.58a, compiled Dec 5 2012 18:38:08
Firmware: J-Link ARM / Flasher ARM V3 compiled Nov 28 2012 18:57:58
Hardware: U3.00
S/N: 163000100
Feature(s): JFlash
IP-Addr.: No configuration received via DHCP
UTarget = 3.267U
Info: TotalIRLen = 4, IRPrint = 0x01
Found 1 JTAG device, Total IRLen = 4:
#0 Id: 0x3F0F0F0F, IRLen: 04, IRPrint: 0x1, ARM7TDMI Core
Found ARM with core Id 0x3F0F0F0F (ARM7)
JTAG speed: 100 kHz
J-Link>_

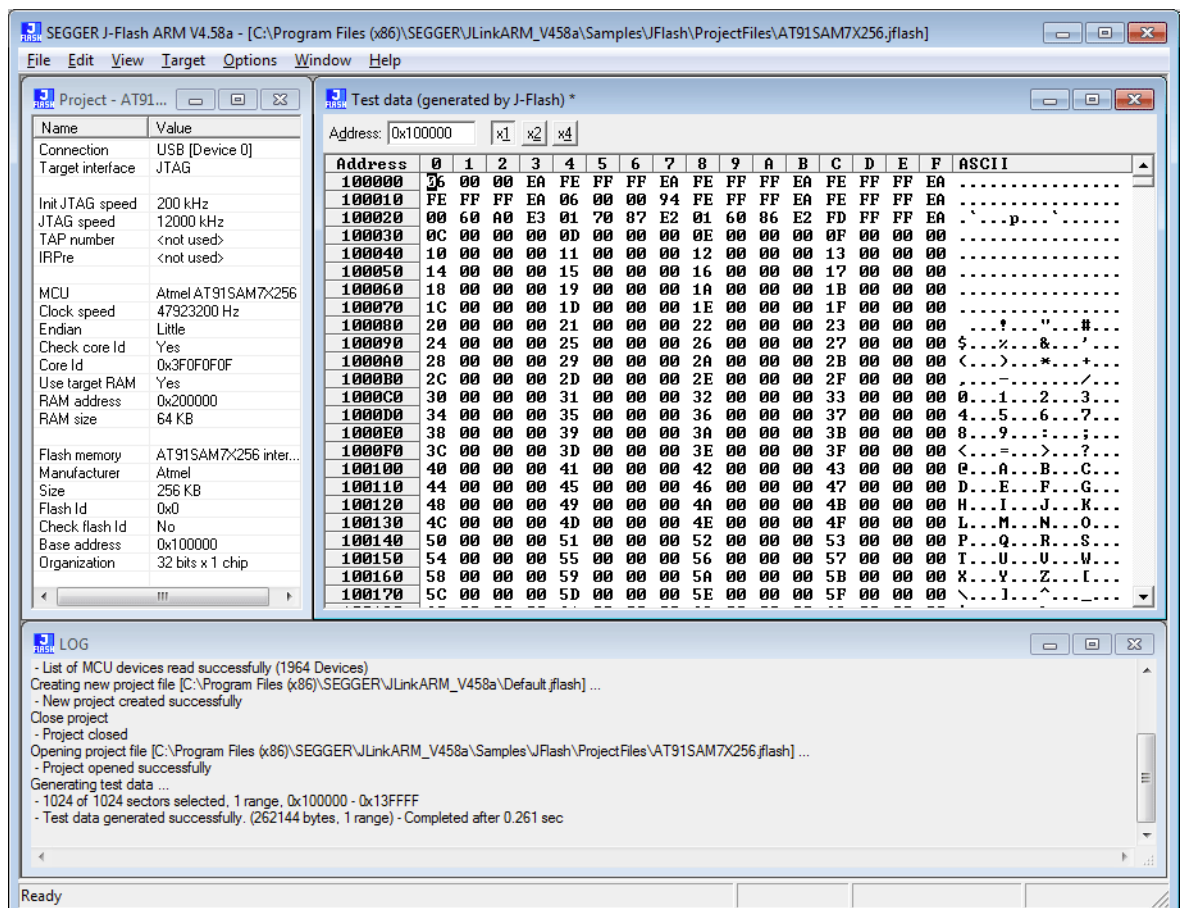
```

2.3.1.2 Setting up Flasher for stand-alone mode

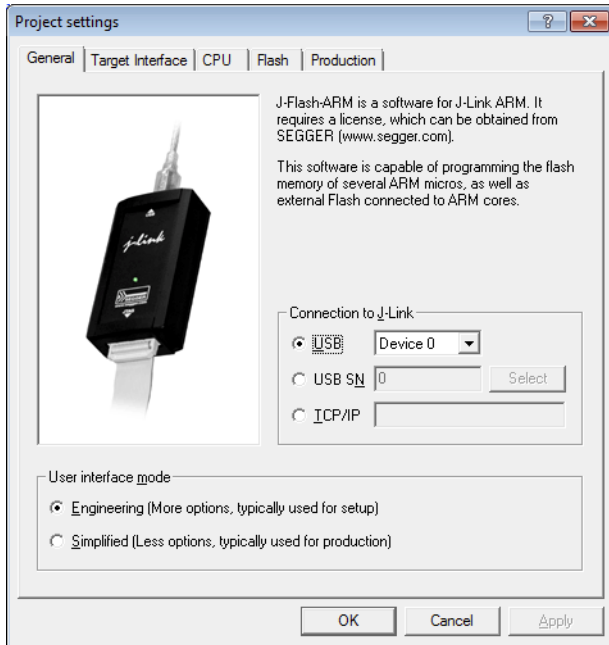
In order to set up Flasher for the stand-alone mode it needs to be configured once using the J-Flash software. For more information about J-Flash, please refer to the *J-Flash User Guide*.

After starting J-Flash, open the appropriate J-Flash project for the target Flasher 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 -> New Project**.

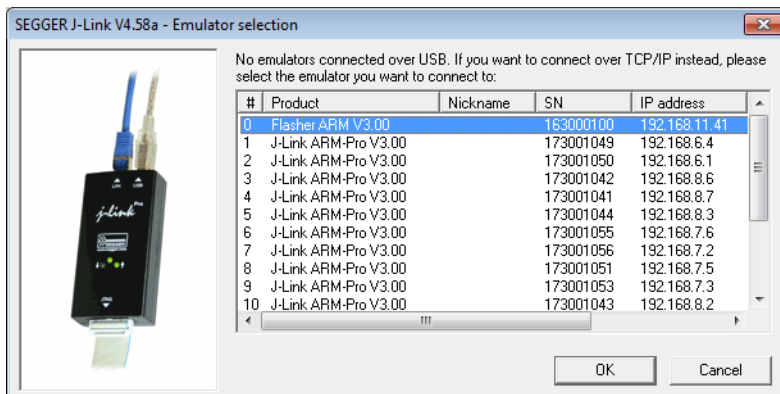
After the appropriate project has been opened / created, the data file which shall be programmed needs to be loaded, by selecting **File -> open**. After this J-Flash should look like in the screenshot below.



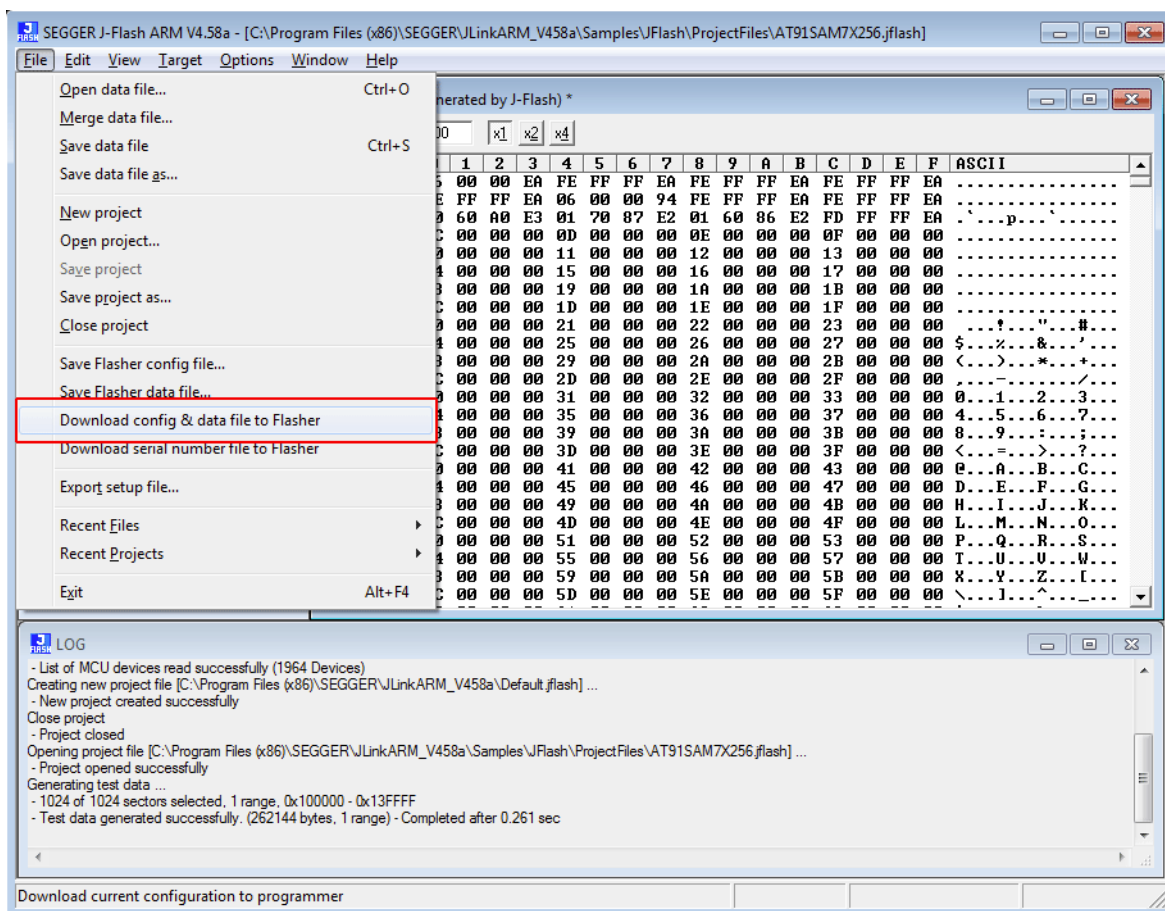
Before downloading the configuration (project) and program data (data file) to Flasher, the connection type (USB/IP) needs to be selected in the project. These settings are also saved on a per-project basis, so this also only needs to be setup once per J-Flash project. The connection dialog is opened by clicking **Options -> Project settings -> General**.



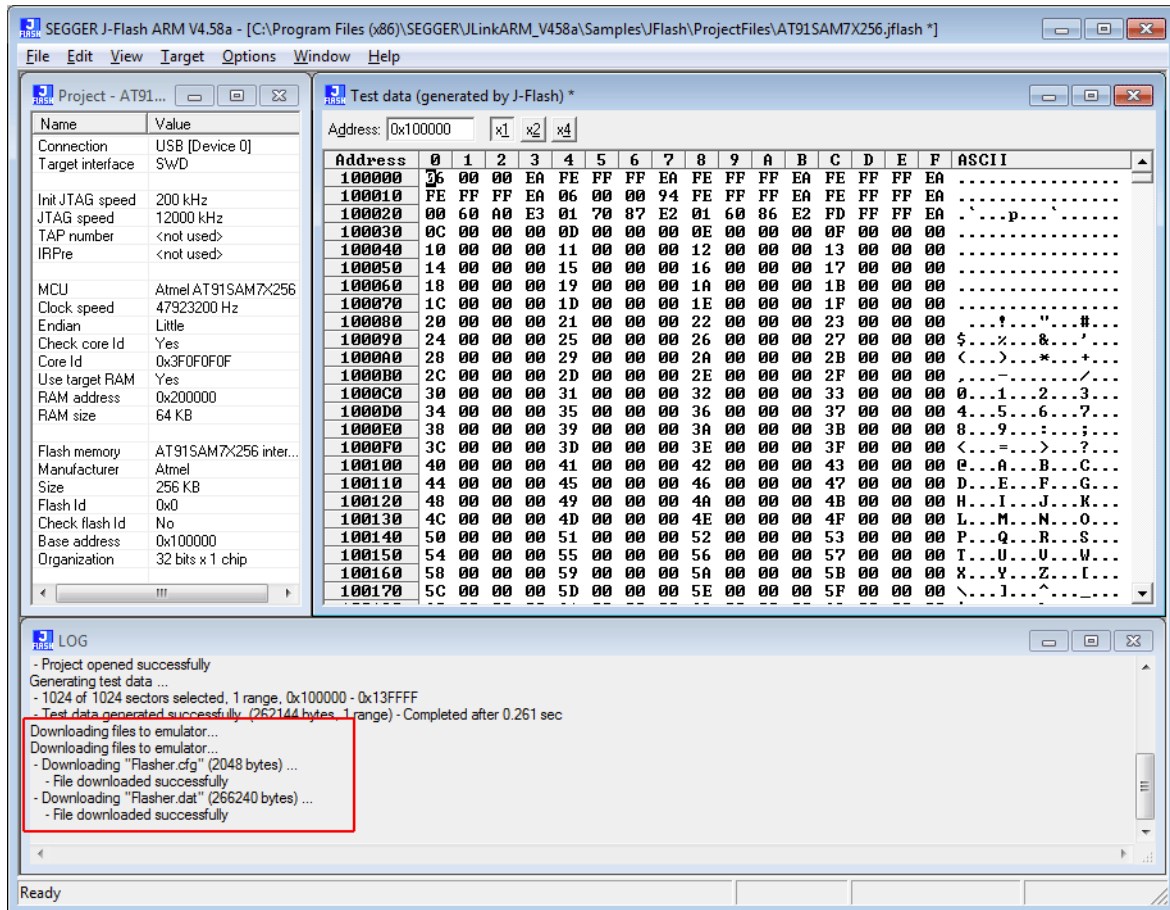
The connection dialog allows the user to select how to connect to Flasher. When connecting to a Flasher via TCP/IP it is not mandatory to enter an IP address. If the field is left blank and **File->Download to programmer** is selected, an emulator selection dialog pops up which shows all Flasher which have been found on the network. The user then can simply select the Flash he wants to download the configuration to.



In order to download the configuration and program data to the Flasher, simply select **File -> Download config & data file to Flasher**.



The J-Flash log window indicates that the download to the emulator was successful.



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

2.3.1.3 LED status indicators

In J-Link mode, there are also certain LED status codes defined:

#	Status of LED	Meaning
0	GREEN high frequency blinking (On/Off time: 50ms => 10Hz)	Flasher is waiting for USB enumeration or ethernet link. As soon as USB has been enumerated or ethernet link has been established, the green LED stops flashing and is switched to constant green.
0	GREEN constant	Flasher enumeration process is complete and it is ready to be controlled by a PC application.

Table 2.1: LED status indicators in Stand-alone mode

2.3.2 Stand-alone mode

In order to use Flasher in "stand-alone mode", it has to be configured first, as described in *Setting up Flasher for stand-alone mode* on page 27. To boot Flasher in the "stand-alone mode", only the power supply to Flasher has to be enabled (Flasher should not be connected to a PC). In the "stand-alone mode" Flasher can be used as a stand-alone flash programmer.

Note: Flasher can only program the target device it was configured for. In order to program another target device, you have to repeat the steps described in *Setting up Flasher for stand-alone mode* on page 27.

2.3.2.1 LED status indicators

Progress and result of an operation is indicated by Flasher's LEDs. The behavior is different for J-Link and stand-alone mode. For a definition of the different modes, please refer to *Operating modes* on page 26.

The following table describes the behavior of the LEDs in stand-alone mode.

#	Status of LED	Meaning
0	GREEN constant	Flasher waits for a start trigger to perform an operation in stand-alone mode.
1	GREEN slow blinking	Flashing operation in progress: <ol style="list-style-type: none"> 1. Erasing (slow blinking on/off time: 80 ms => 6.25 HZ) 2. Programming (slow blinking on/off time: 300ms => ~1.67 Hz) 3. Verifying (slow blinking, on/off time: 100ms => 5 Hz)
2	GREEN: constant RED: off or constant	GREEN constant, RED off: Operation successful. GREEN constant, RED constant: Operation failed Goes back to state #0 automatically, but in case of operation failed, RED remains on until state #1 is entered the next time.

Table 2.2: LED status indicators in stand-alone mode

Older Flasher models have a different behavior. The following serial number ranges behave different from the table above. Flashers with the following serial number ranges behave different and comply to the table below:

- 1621xxxxx (Flasher ARM V2)
- 1630xxxxx (Flasher ARM V3)
- 4210xxxxx (Flasher PPC V1)
- 4110xxxxx (Flasher RX V1)

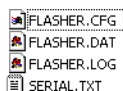
#	Status of LED	Meaning
0	GREEN high frequency blinking (On/Off time: 50ms => 10Hz)	Flasher is waiting for USB enumeration or ethernet link. As soon as USB has been enumerated or ethernet link has been established, the green LED stops flashing and is switched to constant green. In stand-alone-mode, Flasher remains in the high frequency blinking state until state #1 is reached. Flasher goes to state #1 as soon as a #START command has been received via the ASCII interface or the Start button has been pushed.
1	GREEN constant	Connect to target and perform init sequence.
2	GREEN slow blinking	Flashing operation in progress: <ol style="list-style-type: none"> 4. Erasing (slow blinking on/off time: 80 ms => 6.25 HZ) 5. Programming (slow blinking on/off time: 300ms => ~1.67 Hz) 6. Verifying (slow blinking, on/off time: 100ms => 5 Hz)
3	GREEN: constant RED: off or constant	GREEN constant, RED off: Operation successful. GREEN constant, RED constant: Operation failed Goes back to state #0 automatically, but in case of operation failed, RED remains on until state #1 is entered the next time.

Table 2.3: LED status indicators in stand-alone mode (old Flashers)

2.3.3 MSD mode

When pressing the Start/Stop button of Flasher while connecting it to the PC, Flasher will boot in the "MSD mode". This mode can be used to downdate a Flasher firmware version if a firmware update did not work properly and it can be used to configure Flasher for the "stand-alone mode", without using J-Flash.

If Flasher has been configured for "stand-alone mode" as described in the section above, there will be four files on the MSD, `FLASHER.CFG`, `FLASHER.DAT`, `FLASHER.LOG`, `SERIAL.TXT`.



`FLASHER.CFG` contains the configuration settings for programming the target device and `FLASHER.DAT` contains the data to be programmed. `FLASHER.LOG` contains all logging information about the commands, performed in stand-alone mode. The `SERIAL.TXT` contains the serial number, which will be programmed next. J-Flash supports to configure Flasher for automated serial number programming. For further information about how to configure

Currently, J-Flash does not support to configure Flasher for automated serial number programming.

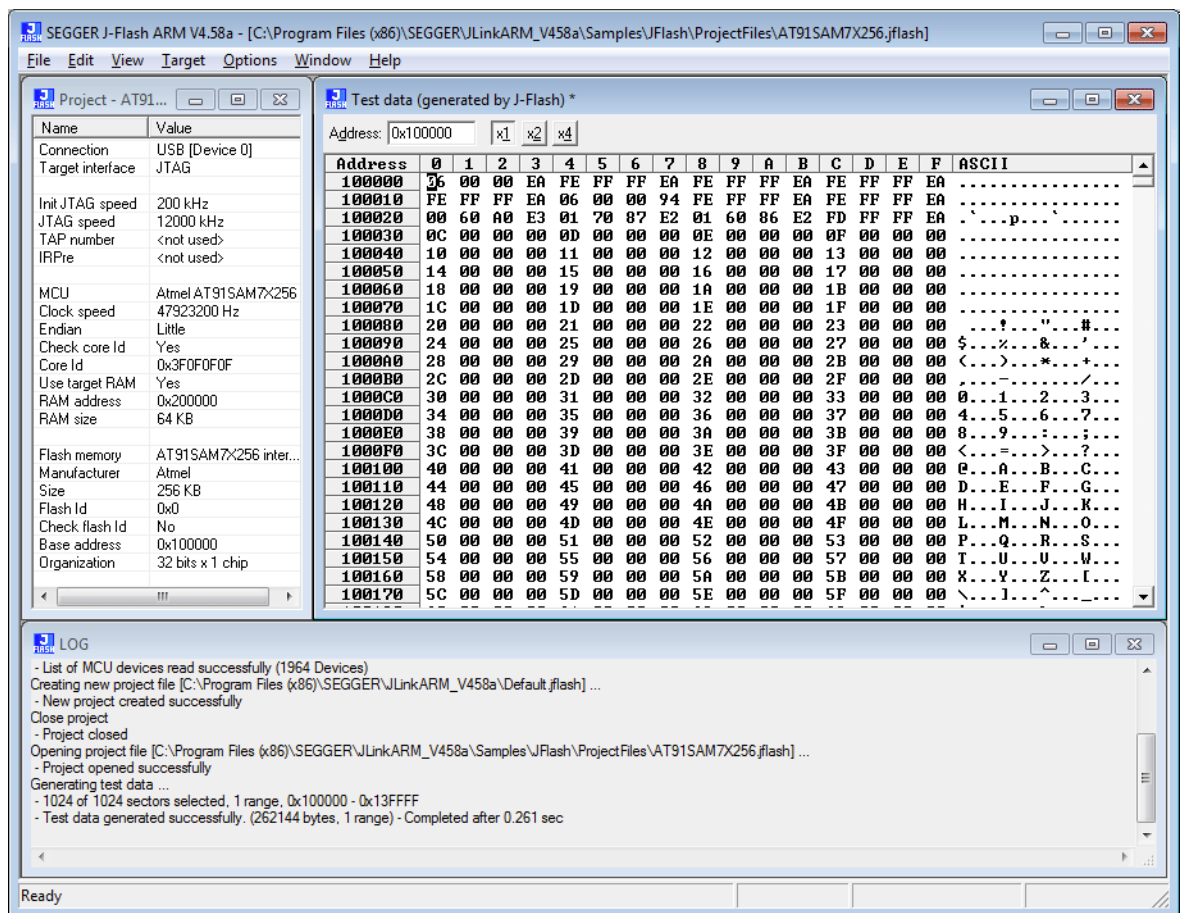
If you want to configure multiple Flasher for the same target you do not have to use J-Flash all the time. It is also possible to copy the `FLASHER.CFG` and the `FLASHER.DAT` files from a configured Flasher to another one. To copy these files boot Flasher in "MSD mode".

2.4 Setting up Flasher for stand-alone mode

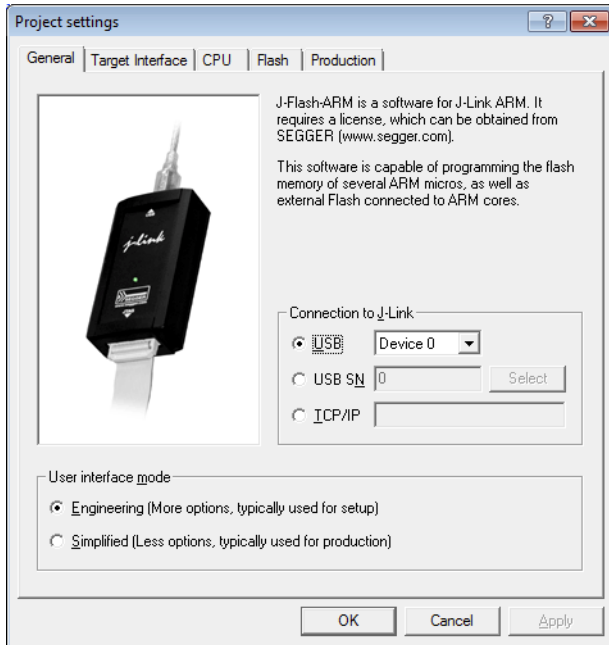
In order to set up Flasher for the stand-alone mode it needs to be configured once using the J-Flash software. For more information about J-Flash, please refer to the *J-Flash User Guide*.

After starting J-Flash, open the appropriate J-Flash project for the target Flasher 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 -> New Project**.

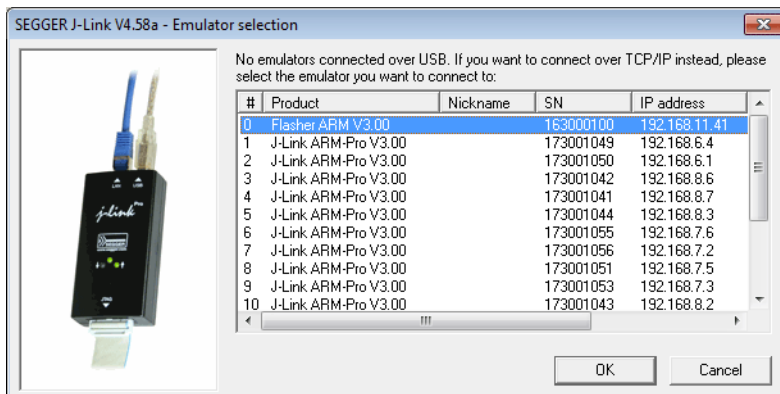
After the appropriate project has been opened / created, the data file which shall be programmed needs to be loaded, by selecting **File -> Open**. After this J-Flash should look like in the screenshot below.



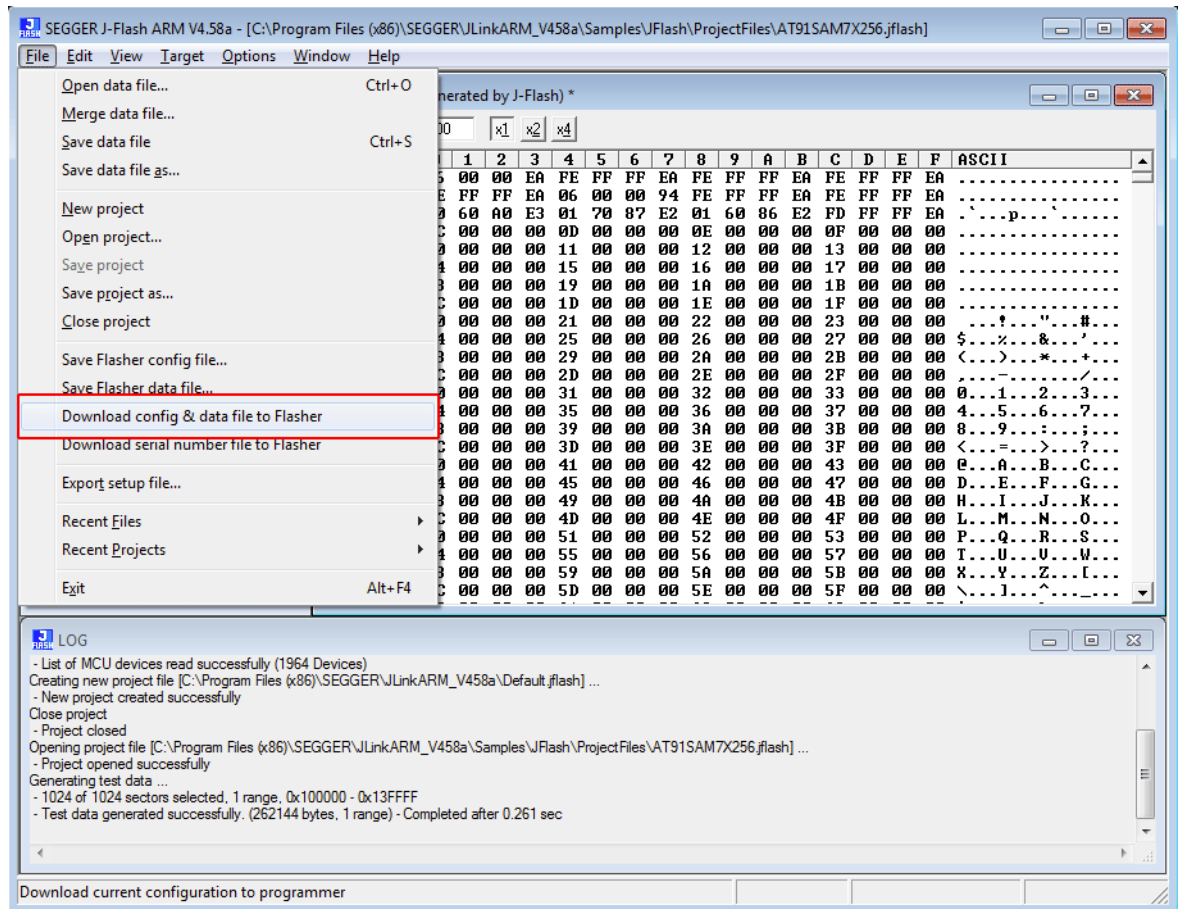
Before downloading the configuration (project) and program data (data file) to Flasher, the connection type (USB/IP) needs to be selected in the project. These settings are also saved on a per-project basis, so this also only needs to be setup once per J-Flash project. The connection dialog is opened by clicking **Options -> Project settings -> General**.



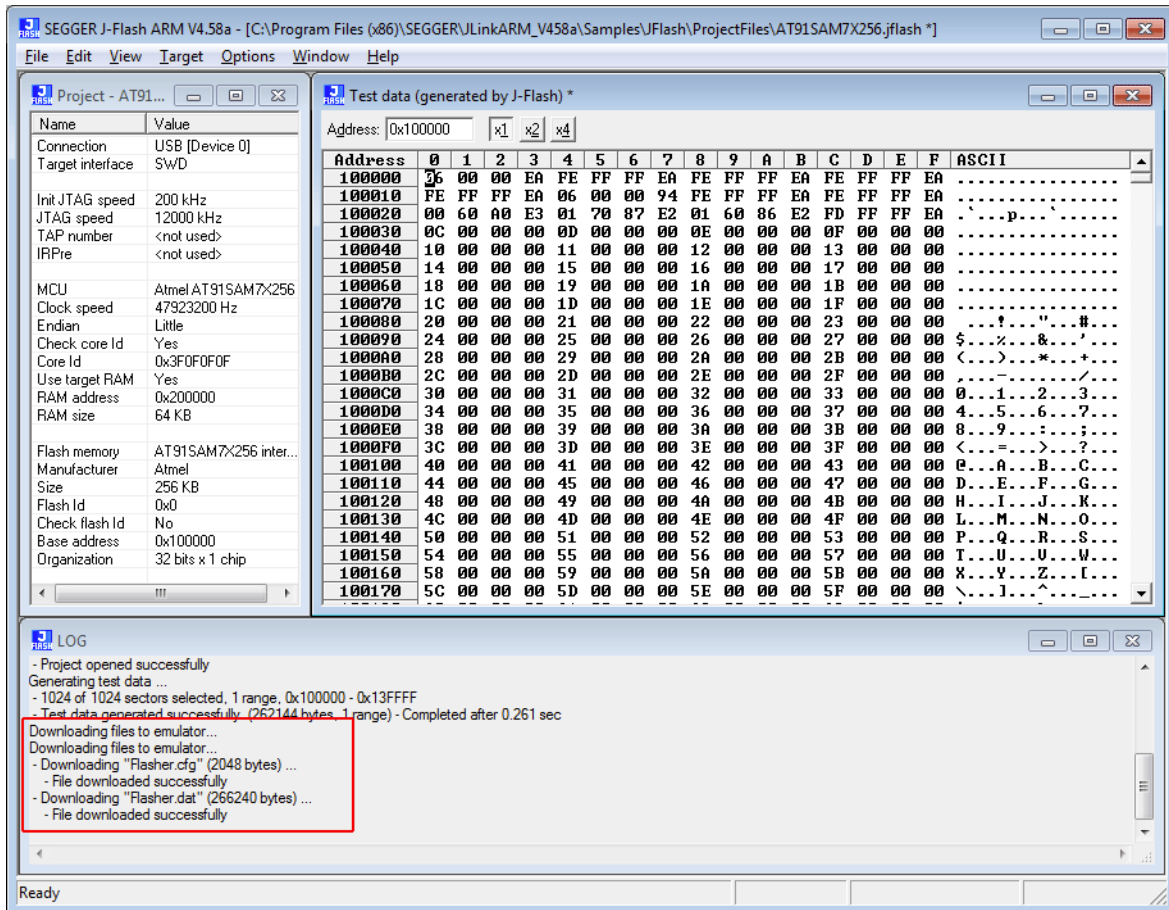
The connection dialog allows the user to select how to connect to Flasher. When connecting to a Flasher via TCP/IP it is not mandatory to enter an IP address. If the field is left blank and **File->Download to programmer** is selected, an emulator selection dialog pops up which shows all Flasher which have been found on the network. The user then can simply select the Flash he wants to download the configuration to.



In order to download the configuration and program data to the Flasher, simply select **File -> Download config & data file to Flasher**.



The J-Flash log window indicates that the download to the emulator was successful.



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

2.4.1 Preparing for stand-alone operation manually

As an alternative, J-Flash can also be used to save config and data file to a hard drive.

This files can later be copied to a Flasher without using J-Flash, which is useful to prepare additional Flasher for stand-alone programming, if for example a company plans to widen its production, new Flasher units can be bought and used in production by simply copying the files to the new units.

Creating config and data files

J-Flash config (*.CFG) and data (*.DAT) files can be created by using the "Save Flasher config file..." and "Save Flasher data file..." options in the "File" menu.

For some devices, additional files (*.PEX) are needed. J-Flash will create a subdirectory (in the same directory as the config file) with the same name as the config file and place the files needed in this directory.

About *.PEX files

When using the "Save Flasher config file..." menu point or when using the "multiple configurations stored on Flasher" feature (See "Multiple File Support" on page 38.) feature, the following needs to be taken care of:

For some devices, special connect, reset etc. sequences are necessary which are stored in so-called *.PEX files on the Flasher.

When using the "Download config & data file to Flasher", J-Flash takes care of correct use and download of these files to Flasher.

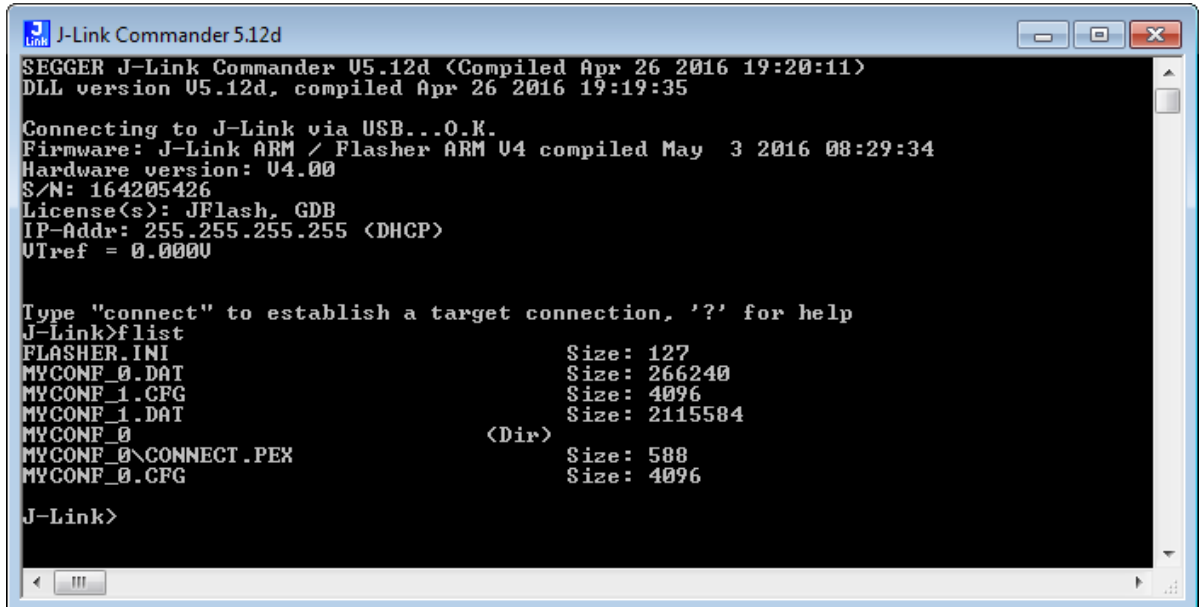
When creating the config files manually and later download them to the Flasher manually, it is user's responsibility to put them at the right place.

The *.PEX files need to be placed in a subdirectory with the same name as the corresponding *.cfg file. J-Flash creates a directory with the correct name automatically when a config file is created.

Example:

MyConf0 is a project for a device that requires a *.PEX file for connect.

MyConf1 is a project for a device that requires no *.PEX file at all.



```

J-Link Commander 5.12d
SEGGER J-Link Commander V5.12d <Compiled Apr 26 2016 19:20:11>
DLL version V5.12d, compiled Apr 26 2016 19:19:35

Connecting to J-Link via USB...O.K.
Firmware: J-Link ARM / Flasher ARM V4 compiled May  3 2016 08:29:34
Hardware version: V4.00
S/N: 164205426
License(s): JFlash, GDB
IP-Addr: 255.255.255.255 <DHCP>
UTref = 0.0000

Type "connect" to establish a target connection, '?' for help
J-Link>flist
FLASHER.INI                Size: 127
MYCONF_0.DAT                Size: 266240
MYCONF_1.CFG                Size: 4096
MYCONF_1.DAT                Size: 2115584
MYCONF_0                    <Dir>
MYCONF_0\CONNECT.PEX        Size: 588
MYCONF_0.CFG                Size: 4096

J-Link>

```

2.5 Multiple File Support

It is also possible to have multiple data files and config files on Flasher, to make Flasher more easy to use in production environment. To choose the correct configuration file and data file pair, a `FLASHER.INI` file is used. This init file contains a `[FILES]` section which describes which configuration file and which data file should be used for programming. A sample content of a `FLASHER.INI` file is shown below:

```
[FILES]
DataFile = "Flasher1.dat"
ConfigFile = "Flasher1.cfg"
```

Using this method all configuration files and data files which are used in the production only have to be downloaded once. From there on a configuration file / data file pair can be switched by simply replacing the `FLASHER.INI` by a new one, which contains the new descriptions for the configuration file and data file. The `FLASHER.INI` can be replaced in two ways:

1. Boot Flasher in MSD mode in order to replace the `FLASHER.INI`
2. If Flasher is already integrated into the production line, runs in stand-alone mode and can not be booted in other mode: Use the file I/O commands provided by the ASCII interface of Flasher, to replace the `FLASHER.INI`. For more information about the file I/O commands, please refer to *File I/O commands* on page 66.

Note: Flasher supports 8.3 filenames only (8 characters filename, 3 characters file extension). Using longer filenames may result in incorrect operation.

2.5.1 Flasher Portable specifics

Flasher Portable allows to choose between four configuration and data file pairs during runtime by using the select/arrow button on the front of Flasher Portable.

Which config / data file pair is used for which image selection position is determined by the contents of the `FLASHER.INI`. For this, the `FLASHER.INI` contents in the `[FILES]` section have been extended. The sample below shows how to enable the user to select between four different images on the Flasher portable via the select / arrow button:

```
[FILES]
DataFile = "First.dat"
ConfigFile = "First.cfg"
DataFile1 = "Second.dat"
ConfigFile1 = "Second.cfg"
DataFile2 = "Third.dat"
ConfigFile2 = "Third.cfg"
DataFile3 = "Fourth.dat"
ConfigFile3 = "Fourth.cfg"
```

Using this method, all configuration files and data files which are used in the production only have to be stored on Flasher Portable via MSD mode. From there on, switching between the files can be done by simply using the selection button of Flasher Portable.

2.5.2 Example

1 Target, 2 Datenfiles (e.g. boot loader und application) --> same configuration file (*.CFG) but different data files (*.DAT) should be used.

- Open pre-configured J-Flash project
- File -> Save Flasher config file ... (DEFAULT.CFG)
- Open data file 1 (boot loader)
- File -> Save Flasher data file ... (BOOT.DAT)
- Open data file 2 (application)
- File -> Save Flasher data file ... (APP.DAT)
- Create the a FLASHER.INI file (content see below)
- Connect the Flasher in MSD mode to the PC
- Copy DEFAULT.CFG, BOOT.DAT, APP.DAT and FLASHER.INI on the Flasher

FLASHER.INI content:

```
[FILES]
DataFile = "BOOT.DAT"
ConfigFile = "DEFAULT.CFG"
DataFile1 = "APP.DAT"
ConfigFile1 = "DEFAULT.CFG"
```

2.6 Programming multiple targets

It is possible to program multiple targets which are located in a JTAG chain. The targets will be programmed each with a configuration and a data file. The configuration for the desired target must be selected before it can be programmed, this can be done with the `#SELECT` command. For more information how to use the `#SELECT` command please refer to Chapter "3.3.5 Commands to Flasher".

Example

Three devices should be programmed.

JTAG Chain: *TDI --> Device2 --> Device1 --> Device0 --> TDO*

Three configurations would be stored on the flasher:

Config 0: Configured to program Device0 (DEVICE0.CFG, DEVICE0.DAT)

Config 1: Configured to program Device1 (DEVICE1.CFG, DEVICE1.DAT)

Config 2: Configured to program Device2 (DEVICE2.CFG, DEVICE2.DAT)

Selection and programming of the target will be done via the ASCII interface:

```
#SELECT DEVICE0
#AUTO
#SELECT DEVICE1
#AUTO
#SELECT DEVICE2
#AUTO
```

2.6.1 Programming multiple targets with J-Flash

Programming multiple targets can also be done via J-Flash using the command line interface. For this each target must be handled with its own project file.

Example

```
JFlash.exe -openproj"Device0.jflash" -open"Device0.hex" -auto -exit
JFlash.exe -openproj"Device1.jflash" -open"Device1.hex" -auto -exit
JFlash.exe -openproj"Device2.jflash" -open"Device2.hex" -auto -exit
```


2.7 Batch Programming in stand-alone mode

Batch programming allows to execute different stand-alone mode jobs in batch to be executed in immediate succession, without any user interaction in between. This can be used for example to program multiple targets in a JTAG-Chain or multiple data files to a target.

A batch may contains an unlimited number of configurations which consist of a data file (*.DAT) and config file (*.CFG). For further information regarding config and data files, please refer to *Preparing for stand-alone operation manually* on page 36).

In order to specify the batch jobs, a FLASHER.INI file is used. This init file contains a [BATCH] section which describes which configuration pairs (*.DAT and *.CFG file) should be used for each batch job.

A sample content of a FLASHER.INI file is shown below:

```
[BATCH]
DataFile = "Flasher0.dat"
ConfigFile = "Flasher0.cfg"
DataFile1 = "Flasher1.dat"
ConfigFile1 = "Flasher1.cfg"
```

Creating / Replacing of the FLASHER.INI file can done in two ways:

1. Boot Flasher in MSD mode in order to replace the FLASHER.INI
2. If Flasher is already integrated into the production line, runs in stand-alone mode and can not be booted in other mode: Use the file I/O commands provided by the ASCII interface of Flasher, to replace the FLASHER.INI. For more information about the file I/O commands, please refer to *File I/O commands* on page 66.

In case of an error occurred during execution, the Flasher terminates the entire batch processing.

Note: Please note that the batch programming feature can not be used with the multiple file support feature. Therefore, neither the #SELECT ASCII command nor the [FILES] tag in the FLASHER.INI file can be used.

Note: Flasher supports 8.3 filenames only (8 characters filename, 3 characters file extension). Using longer filenames may result in incorrect operation.

2.7.1 Flasher Portable specifics

Flasher Portable allows to choose between four different batches during runtime by using the select/arrow button on the front of Flasher Portable. Which batch configuration is used for which image selection position is specified in the FLASHER.INI. For this, the FLASHER.INI contents in the [BATCH] section have been extended. The sample below shows how to enable the user to select between four different batches on the Flasher Portable via the select / arrow button:

```
[BATCH]
DataFile = "Flasher0.dat"
ConfigFile = "Flasher0.cfg"
DataFile1 = "Flasher1.dat"
ConfigFile1 = "Flasher1.cfg"
DataFile2 = "Flasher2.dat"
ConfigFile2 = "Flasher2.cfg"
[BATCH1]
DataFile = "TEST.dat"
ConfigFile = "Test.cfg"
[BATCH2]
DataFile = "VALIDATE.dat"
ConfigFile = "Flasher0.cfg"
```

Using this method allows to have different batches for different setups used in the production to be stored once on the Flasher Portable via MSD mode. From there on, switching between the batches can be done by simply using the selection button of Flasher Portable.

2.7.2 Examples

Example 1: Programming two Data files to the same target

- Open your J-Flash project.
- Use File -> Save Flasher config file... to save the .CFG file (in this example: STM32F4.CFG).
- Select the first binary and use File -> Save Flasher Data file... to save the first .DAT file (in this example: DATA0.DAT).
- Select the second binary and use File -> Save Flasher Data file... to save the second data file .DAT file (in this example: DATA1.DAT).
- Copy the Files to the Flasher e.g. by using MSD mode.
- Create a FLASHER.INI file in the root directory of the Flasher.
- Exemplary content of FLASHER.INI:

```
[BATCH]
DataFile = "DATA0.dat"
ConfigFile = "STM32F4.cfg"
DataFile1 = "DATA1.dat"
ConfigFile1 = "STM32F4.cfg"
```

Example 2: Programming one Data file to the first target in a JTAG-Chain and then programming two data files to another device in the JTAG chain.

Example scenario: 2 Devices in a JTAG chain, a STM32F1 and a STM32F4.

- Follow the same as described before and additionally:
- Create one project file per target (and create a .CFG file of each one).
- Make sure each project file is configured correctly, especially the JTAG-Chain position (See UM8003 "J-Flash" for more detailed info).
- Exemplary content of FLASHER.INI:

```
[BATCH]
DataFile = "F1DATA.dat"
ConfigFile = "STM32F1.cfg"
DataFile1 = "F4DATA0.dat"
ConfigFile1 = "STM32F4.cfg"
DataFile2 = "F4DATA1.dat"
ConfigFile2 = "STM32F4.cfg"
```

Example 3: Using multiple Batch sections with Flasher Portable.

Example scenario: 2 Devices in a JTAG chain, a STM32F1 and a STM32F4.

Selection 1 will program the STM32F1 target.

Selection 2 will program the STM32F4 target using "F4DATA0.dat".

Selection 3 will program the STM32F4 target using "F4DATA1.dat".

Selection 4 will execute 1, 2 and 3 in sequence.

- Exemplary content of FLASHER.INI:

```
[BATCH]
DataFile = "F1DATA.dat"
ConfigFile = "STM32F1.cfg"
[BATCH1]
DataFile = "F4DATA0.dat"
ConfigFile = "STM32F4.cfg"
[BATCH2]
DataFile = "F4DATA1.dat"
ConfigFile = "STM32F4.cfg"
[BATCH3]
DataFile = "F1DATA.dat"
ConfigFile = "STM32F1.cfg"
DataFile1 = "F4DATA0.dat"
ConfigFile1 = "STM32F4.cfg"
DataFile2 = "F4DATA1.dat"
ConfigFile2 = "STM32F4.cfg"
```

2.8 Serial number programming

Flasher 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 (depending on the configuration) need to be configured first.

In general, Flasher 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.

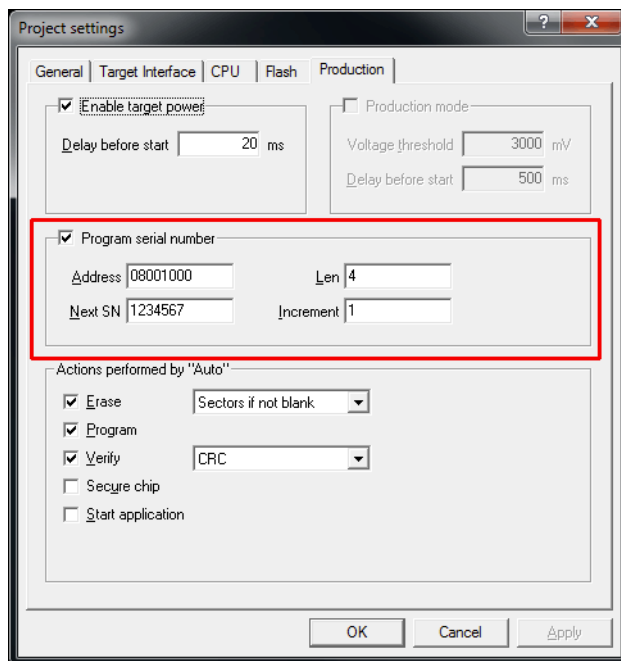
In the following some generic information how to setup Flasher & the J-Flash project for serial number programming are given.

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.

Note: Currently, programming of serial numbers is only supported for stand-alone mode. Future versions of J-Flash may also support serial number programming in J-Link mode.

2.8.1 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	<p>The length of the serial number (in bytes) which should be programmed.</p> <p>If no serial number list file is given, J-Flash allows to use a 1-4 byte serial number. In case of 8 is selected as length, the serial number and its complementary is programmed at the given address.</p> <p>In case a serial number list file is given, Flasher 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.</p>
Next SN	<p>In case no serial number list file is given, Next SN is next serial number which should be programmed. The serial number is always stored in little endian format in the flash memory.</p> <p>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 starts counting with line 0, so in order to start serial number programming with the first line of the <code>SNList.txt</code>, Next SN needs to be set to 0.</p>
Increment	Specifies how much Next SN is incremented.

Table 2.4: Flasher serial number settings

2.8.2 Serial number file

When selecting **File -> Download serial number file to Flasher**, J-Flash will create a Serial number file named as `<JFlashProjectName>_Serial.txt`. This file is downloaded as `SERIAL.TXT` on Flasher. The file is generated based on the serial number settings in the J-Flash project and will contain the value defined by the **Next SN** option. The serial number file can also be manually edited by the user, since the serial number is written ASCII encoded in the `SERIAL.TXT` file.

2.8.3 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.

When selecting **File-> Download serial number file to Flasher**, J-Flash will look for a serial number list file named as <JFlashProjectName>_SNList.txt in the directory where the J-Flash project is located. This file is downloaded as SNList.txt on Flasher. The serial number list file needs to be created manually by the user and has the following syntax:

- One serial number per line
- Each byte of the serial number is described by two hexadecimal digits.

Example

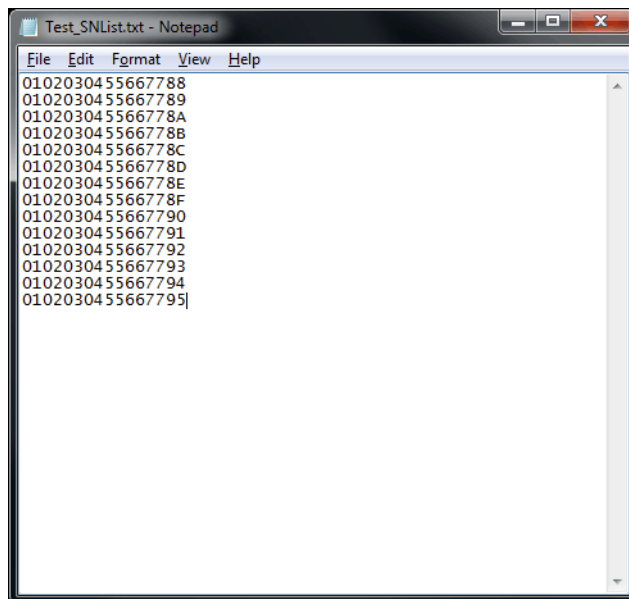
A 8-byte serial number should be programmed at address 0x08000000.

It should be programmed as follows in the memory:

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

The serial number list 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 43.

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 43. 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 filled with 0s by Flasher.

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.

2.8.4 Programming process

Flasher 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 on the Flasher 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 in case if serial number programming is disabled, to make sure that for the Flasher logfile there is a reference which programming cycle passed and which not. As long as serial number programming has not been enabled in the J-Flash project, Flasher does not merge any serial number data into the image data to be programmed.

2.8.5 Downloading serial number files to Flasher

Downloading the serial number files needs to be done explicitly by selecting **File->Download serial number file to Flasher**. Please note that the **File -> Download config & data file to Flasher** option does only download the configuration and data file to Flasher since usually the current serial number used for programming shall not be reset/overwritten when just updating the image Flasher shall program.

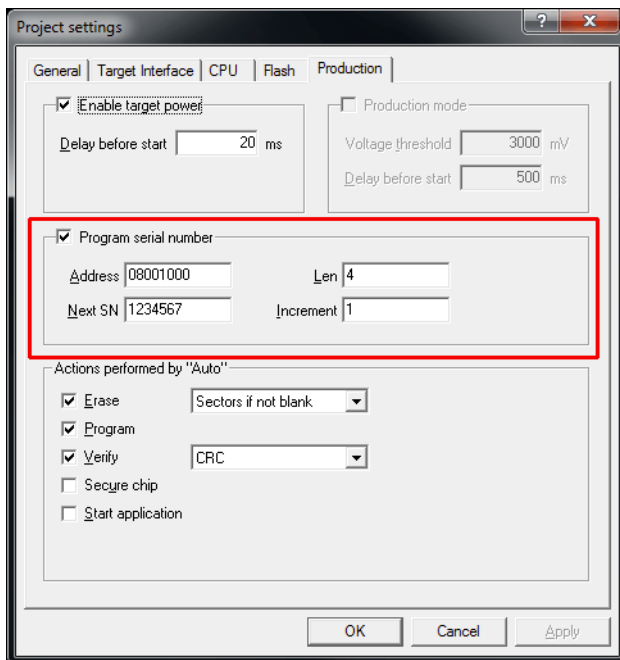
2.8.6 Sample setup

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

Defining serial number address, length and start value

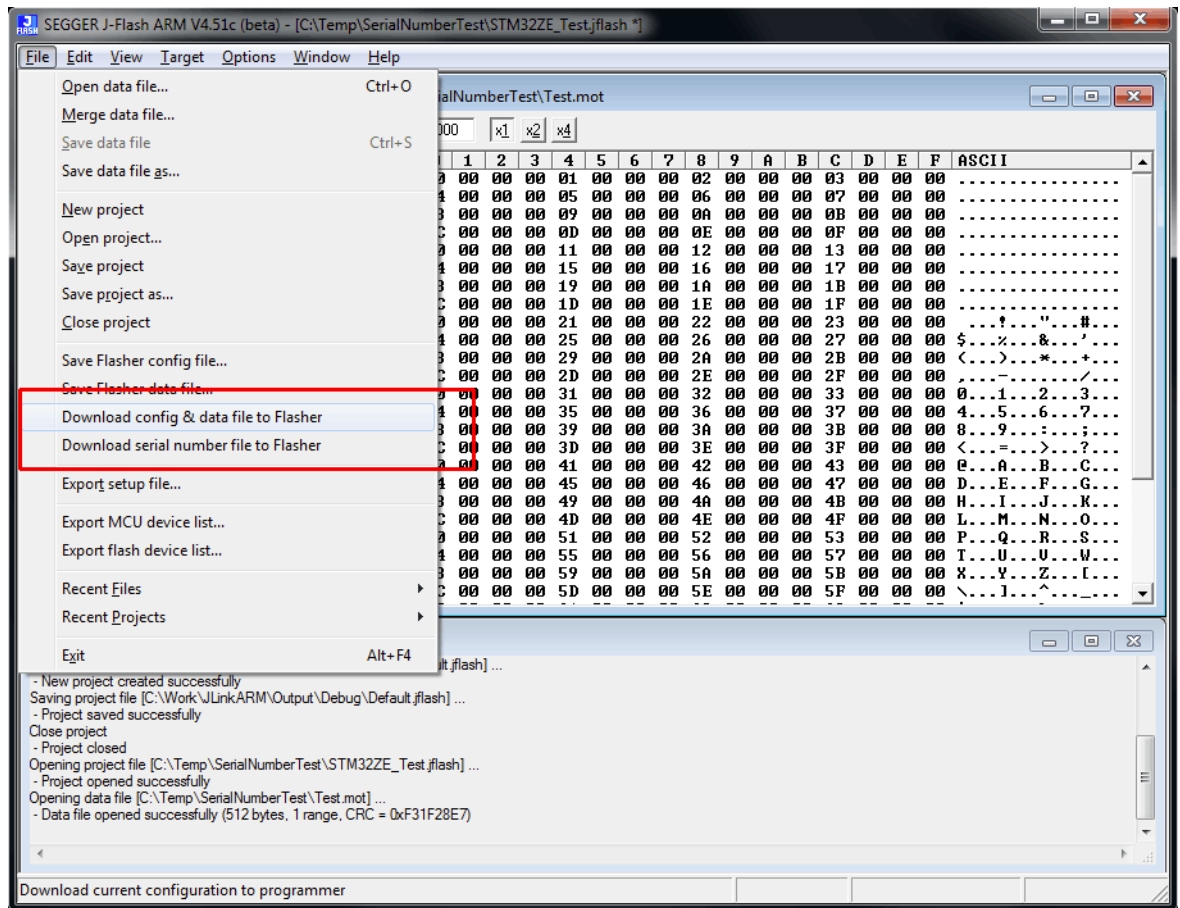
In the J-Flash project the following needs to be defined:

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



Downloading configuration, data and serial number to Flasher.

After setting up the rest of the configuration (Target interface etc.) and selecting an appropriate data file, the configuration, data and serial number file is downloaded into Flasher via the **File -> Download config & data file to Flasher** and **File-> Download serial number file to Flasher** option.



After downloading the serial number to Flasher, J-Flash also created the `<JFlashProjectName>_Serial.txt`.

Share with ▾		New folder	
Name	Date modified	Type	Size
STM32ZE_Test.jflash	25/06/2012 19:34	JFLASH File	3 KB
STM32ZE_Test_Serial.txt	26/06/2012 13:57	Text Document	1 KB
Test.mot	25/06/2012 19:32	MOT File	2 KB

Now Flasher is prepared to program the 8-byte serial number.

2.9 Patch file support

In stand-alone mode Flasher supports patch files which allows to patch the content of the data to be programmed. Before starting programming process in stand-alone mode, Flasher will look for a file named `Patches.txt` being present on the Flasher. 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

The syntax for `<NumPatches>==4` is as follows:

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

Each patch-line contains `<Addr>,<NumBytes>:<Data>`. Find below a table which describes each parameter.

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

Note: All values are expected in hexadecimal format (hex).
`<Data>` section is always preceeded 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 force 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

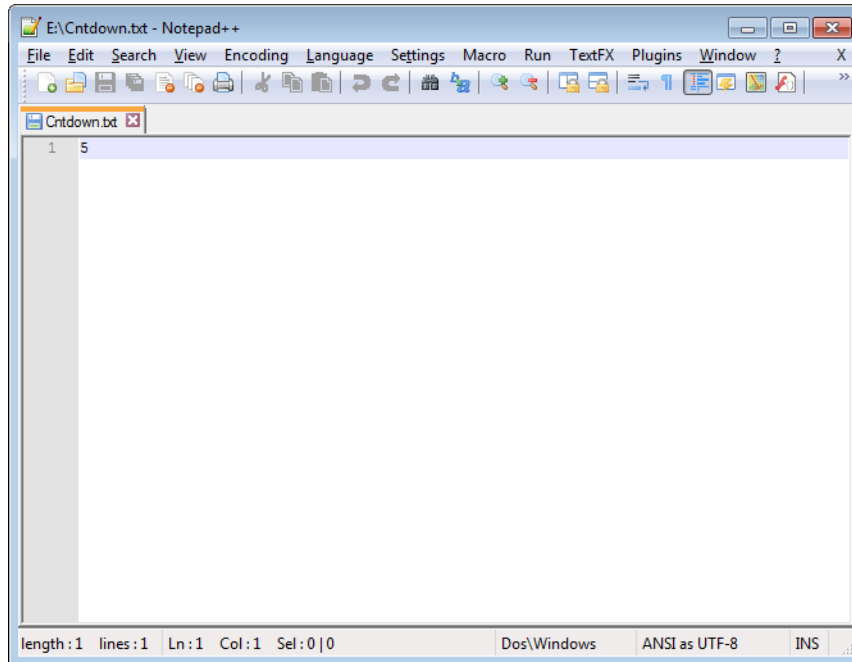
Alternatively, you can start a programming cycle with patch data that is only valid for this one cycle (no need for a `Patches.txt` file):

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

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

2.10 Limiting the number of programming cycles

Flasher 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. To make use of this feature, a file called `Cntdown.txt` needs to be placed on the Flasher. This file simply contains a decimal number (32-bit unsigned integer) that describes how many programming cycles can be performed with the current setup.



This feature especially makes sense when used in combination with authorized flashing. For more information about authorized flashing, please refer to *Authorized flashing* on page 50.

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`.

2.10.1 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 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, ...)

2.10.2 Required Flasher hardware version for `Cntdown.txt` support

Older Flasher models do not support the limiting of programming cycles. The Flashers with the following serial number ranges do not support limiting of programming cycles:

- 1621xxxxx (Flasher ARM V2)
- 1630xxxxx (Flasher ARM V3)
- 4210xxxxx (Flasher PPC V1)
- 4110xxxxx (Flasher RX V1)

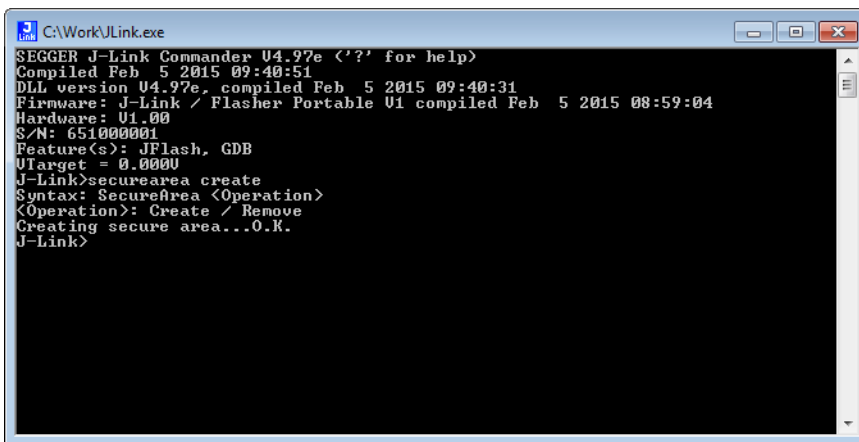
All other models / hardware versions support limiting of programming cycles.

2.11 Authorized flashing

Current hardware versions of Flasher support creation of a so called `secure area` which allows to pre-configure the Flasher with a given setup and then give it to external production facilities etc. without the possibility to read out the Flasher contents via MSD, FILE I/O functionality (J-Link Commander) or RS232 commands. This section describes how to setup a `secure area` on a Flasher and how to move the configuration/data file(s) into it.

2.11.1 Creating / Adding the secure area

By default, Flashers are shipped with a public area only (full Flasher flash size accessible via MSD etc.). The `secure area` has to be activated / created once, to make use of it. This will reserve half of the Flasher storage size (on current models this will be ~64 MB) for the `secure area`. The `secure area` can be removed at any time, providing the full flasher storage to the public area again. The `secure area` can be created / removed via J-Link Commander, which is part of the software package that comes with Flasher.



```
C:\Work\JLink.exe
SEGGER J-Link Commander V4.97e ('?' for help)
Compiled Feb  5 2015 09:40:51
DLL version V4.97e, compiled Feb  5 2015 09:40:31
Firmware: J-Link / Flasher Portable V1 compiled Feb  5 2015 08:59:04
Hardware: V1.00
S/N: 651000001
Feature(s): JFlash, GDB
VTarget = 0.0000
J-Link>securearea create
Syntax: SecureArea <Operation>
<Operation>: Create / Remove
Creating secure area...O.K.
J-Link>
```

The following `secure area` related commands are available in J-Link Commander:

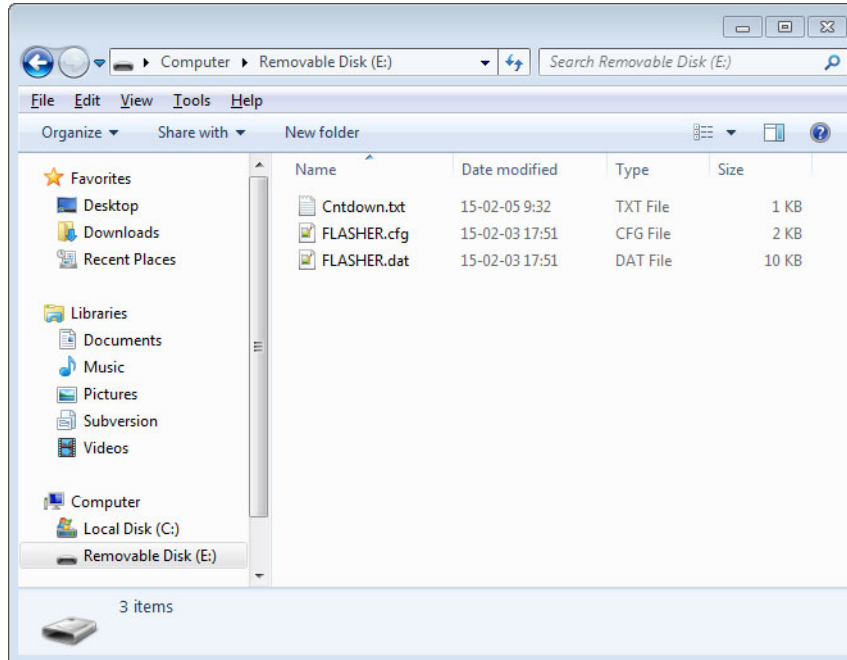
- `securearea create`
- `securearea remove`

Note: When creating or removing the `secure area`, all configuration and data files being stored on the Flasher, are lost. Please make sure that they are not needed anymore, before adding / removing the security area.

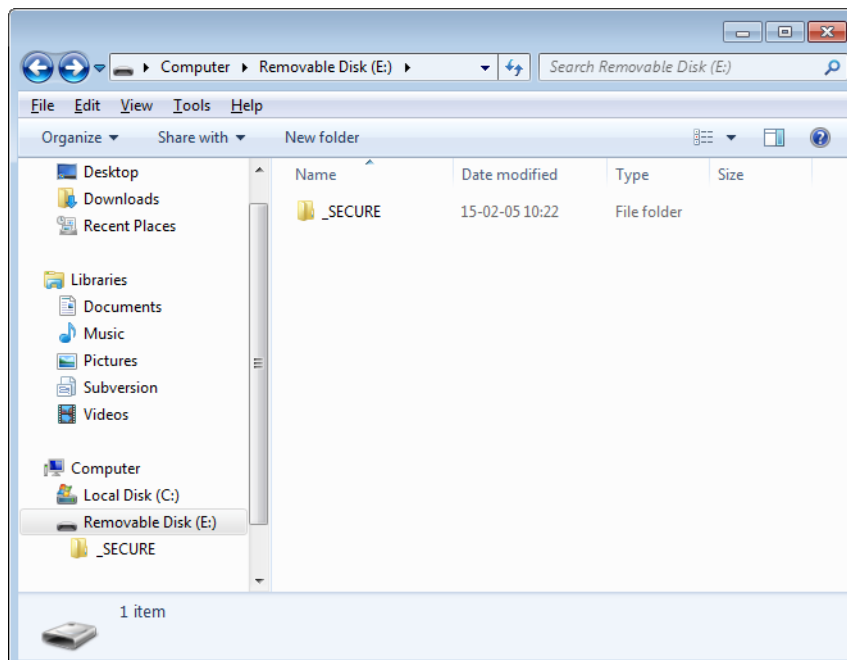
2.11.2 Moving files to the secure area

Before moving configuration + data to the `secure` area, proper functionality of the setup should be tested in stand-alone mode. Once the setup is working as expected, do the following, to move the configuration + data into the `secure` area:

- Start Flasher in MSD mode (For more info, please refer to: *MSD mode* on page 32)



- Create a folder "_SECURE"
- Move all files that shall be moved to the `secure` area, into this folder



- Reboot Flasher (Do not enter MSD mode again, yet! Otherwise, contents will not be moved). Now, depending on the configuration and data file size, it may take a bit, before the Flasher Power LED lit. Once it lit, all contents have been moved to the `secure` area and the `_SECURE` folder in the public area has been deleted.
- Now Flasher can be used in stand-alone mode, as normal, but the files cannot be read back by the user / operator.

2.11.3 Considerations to be taken when using the secure area

When using the `secure area`, some things need to be considered:

- All features like multiple file support, patch file support etc. can also be used when operating from the `secure area`.
- The `secure area` cannot be read back by any utility. Solely the `FLASHER.LOG` is always placed and updated in the public area, even when Flasher operates from the `secure area`.
- If there is any file/folder in the public area, except the `FLASHER.LOG` and there is also any configuration / data present in the `secure area`, stand-alone flashing will fail because it is not unambiguous which configuration / data shall be used. In such cases, Flashers with Ethernet / RS232 interface will output an appropriate error message on programming. All Flasher models will output an appropriate error message in the `FLASHER.LOG`.
- Moving files from the public into the secure area can be done multiple times, as explained in *Moving files to the secure area* on page 51. Each time files are moved from the public area to the `secure area`, all contents of the `secure area` are erased first, to make sure that no previous configuration is present there.

2.11.4 Required Flasher hardware version

Older Flasher models do not support authorized flashing. The Flashers with the following serial number ranges do not support authorized flashing:

- 1621xxxxx (Flasher ARM V2)
- 1630xxxxx (Flasher ARM V3)
- 4210xxxxx (Flasher PPC V1)
- 4110xxxxx (Flasher RX V1)

All other models / hardware versions support authorized flashing.

2.12 Target interfaces

The table below shows the supported target interfaces of the different Flasher models.

Hardware	Supported interfaces
Flasher ARM	JTAG, SWD
Flasher RX	JTAG
Flasher PPC	JTAG

Table 2.5: Supported target interfaces by Flasher

For more information about the target interfaces itself, please refer to:

- *UM08001, chapter "Working with J-Link and J-Trace", section "JTAG interface"*
- *UM08001, chapter "Working with J-Link and J-Trace", section "SWD interface"*

2.13 Supported microcontrollers

2.13.1 Flasher

Flasher supports download into the internal flash of a large number of microcontrollers. The number of supported devices is steadily growing, so you can always find the latest list of supported devices on our website:

<http://www.segger.com/supported-devices.html>

2.14 Support of external flashes

2.14.1 Flasher ARM

In general Flasher ARM supports programming of external flashes listed below:

- parallel NOR flash
- serial NOR flash
- NAND flash
- DataFlash

If the parallel NOR flash device which is used is not CFI-compliant you have to select the flash device in J-Flash explicitly, for a list of all parallel NOR flash devices which can be explicitly selected in J-Flash, please refer to *UM08003, J-Flash User Guide*, chapter *Supported Flash Devices*. For serial NOR flash, NAND flash and DataFlash devices a custom RAMCode is needed since the connection of the flash to the CPU differs from device to device. The J-Flash software comes with sample projects for custom RAMCodes. For a complete list of all custom RAMCode projects which come with the J-Flash software, please refer to: <http://www.segger.com/supported-devices.html>

2.14.2 Flasher RX

Programming of external parallel NOR flash is currently not supported by Flasher RX. This limitation will be lifted in the near future.

2.14.3 Flasher PPC

Programming of external parallel NOR flash is currently not supported by Flasher PPC. This limitation will be lifted in the near future.

2.15 Supported cores

2.15.1 Flasher ARM

Flasher ARM supports and has been tested with the following cores, but should work with any ARM7/9, Cortex-M0/M1/M3/M4 core. If you experience problems with a particular core, do not hesitate to contact Segger.

- ARM7TDMI (Rev 1)
- ARM7TDMI (Rev 3)
- ARM7TDMI-S (Rev 4)
- ARM920T
- ARM922T
- ARM926EJ-S
- ARM946E-S
- ARM966E-S
- Cortex-M0
- Cortex-M1
- Cortex-M3
- Cortex-M4

2.15.2 Flasher RX

Flasher RX supports and has been tested with the following cores. If you experience problems with a particular core, do not hesitate to contact Segger.

- RX610
- RX621
- RX62N
- RX62T

2.15.3 Flasher PPC

Flasher PPC supports and has been tested with the following cores. If you experience problems with a particular core, do not hesitate to contact Segger.

- e200z0

2.16 Newline encoding

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

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

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

2.17 Programming multiple targets in parallel

To program multiple targets in parallel, multiple Flasher will be needed (each connecting to one CPU). This can be done using the production programming software J-Flash. For further information, please refer to Chapter "*Command Line Interface*" Sub-chapter "*Programming multiple targets in parallel*" of the J-Flash User Guide (UM08003_JFlash.pdf) which is part of the J-Link Software and Documentation package.

Chapter 3

Remote control

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

3.1 Overview

There are 3 ways to control Flasher operation:

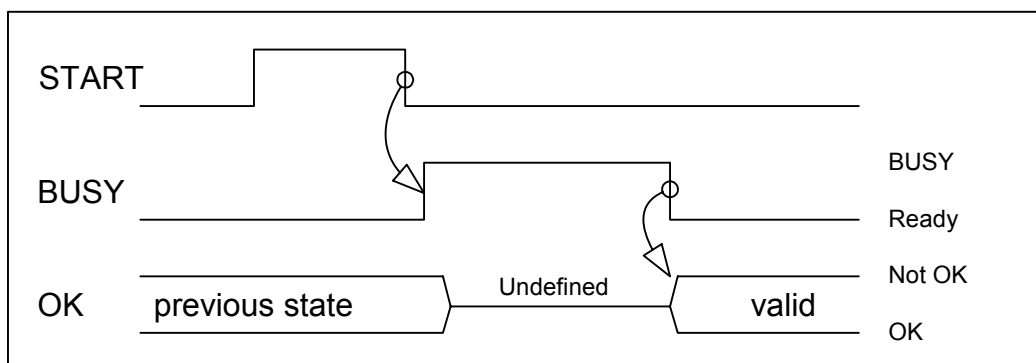
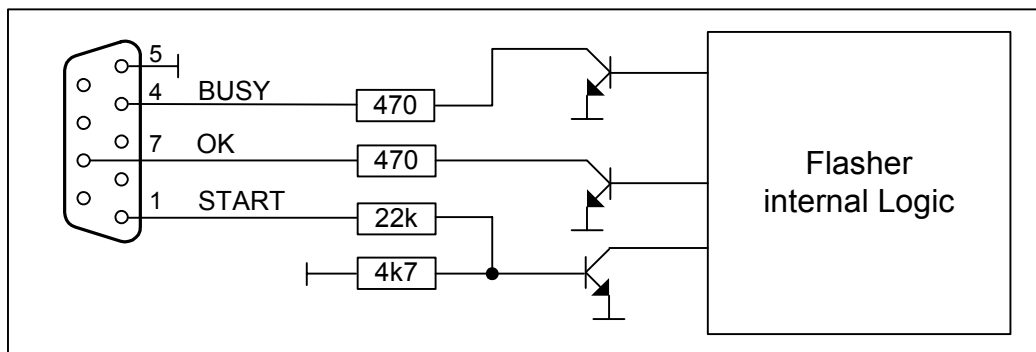
- Manual: Programming operation starts when pressing the button. The LEDs serve as visible indication.
- Via Handshake lines: 3 lines on the serial interface are used.
 - 1 line is an input and can be used to start operation,
 - 2 lines are outputs and serve as Busy and status output
- Terminal communication via RS232.

Note: All three ways to control Flasher operation are working only if Flasher is in standalone mode. In J-Link / MSD mode they have no effect.

3.2 Handshake control

Flasher can be remote controlled by automated testers without the need of a connection to PC and Flasher's PC program. Therefore Flasher 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.

Table 3.1: Flasher LED status

3.3 ASCII command interface

3.3.1 Introduction

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

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

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

3.3.2 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

3.3.3 General usage

Reply messages must be considered in each case. It is essential not to send a new command before a reply for the last one has been received.

3.3.4 Settings for ASCII interface via RS232

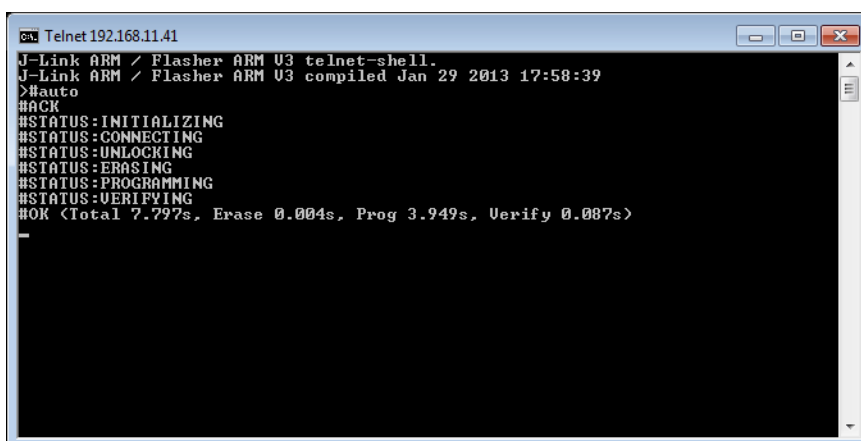
Flasher is driven via a RS232 serial port with the following interface settings:

- 8 data bits,
- no parity
- 1 stop bit

at 9600 baud.

3.3.5 Settings for ASCII interface via Telnet

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



```
ca. Telnet192.168.11.41
J-Link ARM / Flasher ARM U3 telnet-shell.
J-Link ARM / Flasher ARM U3 compiled Jan 29 2013 17:58:39
>#auto
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#STATUS:PROGRAMMING
#STATUS:VERIFYING
#OK (Total 7.797s, Erase 0.004s, Prog 3.949s, Verify 0.087s)
```

3.3.6 Commands to Flasher

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

Commands
#AUTO
#AUTO PATCH
#AUTO NOINFO
#CANCEL
#ERASE
#PROGRAM
#RESULT
#SELECT <Filename>
#START
#STATUS
#VERIFY
File I/O commands
#FCLOSE
#FCRC
#FDELETE <Filename>
#FOPEN <Filename>
#FREAD <Offset>,<NumBytes>
#FSIZE
#FWRITE <Offset>,<NumBytes>:<Data>
#FLIST
#MKDIR <Dirname>
Reply from Flasher
#ACK
#NACK
#OK
#OK:<NumBytes>:<Data>
#OK:<Size>
#STATUS:
#ERRxxx

#AUTO

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

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

- Flasher starts erasing
- Flasher programs target CPU
- Flasher verifies target CPU

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

Finally, Flasher 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.

During execution of the #AUTO command, Flasher automatically sends "status" messages via RS232 to reflect the state of execution. Typically during execution of #AUTO command, Flasher will reply the following sequence of messages:

```
#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)
```

#AUTO PATCH

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

Flasher 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 48.

#AUTO NOINFO

This command may be used instead of #AUTO, if no status messages from Flasher should be sent during execution. The NOINFO extension is also available for all other commands.

The command ends with #OK or #ERRxxx

#BAUDRATE<Baudrate>

This command can be sent in order to change the baudrate of the UART used for the ASCII command interface communication. <Baudrate> is expected in decimal format.

If this command succeeds, Flasher responds with:

```
#ACK
#OK
```

Otherwise it will respond with one of the following error messages:

```
#ERR255: Invalid parameters
```

or

```
#ERR255: Baudrate is not supported
```

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

#CANCEL

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

Flasher will respond with:

```
#ERR007:CANCELED.
```

#ERASE

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

Flasher will reply the following sequence of messages:


```
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#OK (Total 0.893s, Erase 0.483s)
```

#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.

#RESULT

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

#SELECT <Filename>

The #SELECT command is used to select a specific config and data file pair which should be used by Flasher to program the target. <Filename> specifies the name of file pair without extensions (.CFG and .DAT) on the Flasher which should be selected. Flasher saves the selected config and data file in the FLASHER.INI file. So this selection is remembered even between power-cycling Flasher.

This may be very helpful in cases where several config and data files are stored on Flasher. The user can easily switch between these config and data files without connecting Flasher to a host.

If this command succeeds, Flasher responds with:

```
#ACK
#OK
```

Find below a sample sequence which shows how to use the #SELECT command:

```
#SELECT ATSAM7_1                // ATSAM7_1.CFG and ATSAM7_1.DAT is selected
#ACK
#OK
#AUTO                          // Start auto programming
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#STATUS:PROGRAMMING
#STATUS:VERIFYING
#OK (Total 8.416s, Erase 0.005s, Prog 6.845s, Verify 0.959s)

#SELECT ATSAM7_2                // ATSAM7_2.CFG and ATSAM7_2.DAT is selected
#ACK
#OK
#AUTO                          // Start auto programming
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#STATUS:PROGRAMMING
#STATUS:VERIFYING
#OK (Total 8.632s, Erase 0.005s, Prog 7.051s, Verify 0.969s)
```

#START

This command can be sent to release Flasher's target interface. All signals from Flasher to target will be set into high-Z mode, reset of target will be released. It may be used to start target application program.

Flasher will reply with the following sequence of messages:

```
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#OK (Total 1.148s)
```

#STATUS

This command can be sent any time, even during other command execution. Flasher responds with its current state. All defined state messages are described under *Reply from Flasher* on page 68.

#VERIFY

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

3.3.6.1 File I/O commands

The ASCII interface of Flasher also supports file I/O operations via RS232. The following file I/O commands are supported:

#FCLOSE

The #FCLOSE command closes the file on Flasher which was opened via #FOPEN. After this command has been issued further file I/O operations except #FDELETE are not allowed until the #FOPEN command is sent again.

A typical sequence when using the #FCLOSE command does look like as follows:

```
#FCLOSE
#ACK
#OK
```

Note: When using the #FCLOSE command a file has to be open (previously opened by #FOPEN). Otherwise Flasher will respond with the following if no file has been opened:

```
#ACK
#ERR255:No file opened
```

#FCRC

The #FCRC command calculates a 32-bit CRC of the given file. This CRC can be used to verify file integrity. This command should not be used while a file has been opened via #FOPEN. The CRC will be also reported by J-Flash when downloading or saving files via J-Flash.

A typical sequence when using the #FCRC command does look like as follows:

```
#FCRC flasher.dat
#ACK
#OK:0x75BC855A
```

#FDELETE <Filename>

The #FDELETE command is used to delete a file on Flasher where <Filename> specifies the name of the file.

A typical sequence when using the #FDELETE command does look like as follows:

```
#FDELETE flasher.dat
#ACK
#OK
```

Note: If deletion of the file fails for example if the file does not exist, Flasher will respond with the following sequence:

```
#ACK
#ERR255:Failed to delete file
```

#FOPEN <Filename>

The #FOPEN command is used to open a file on Flasher for further file I/O operations. <Filename> specifies the file on the Flasher which should be opened. If <Filename> can not be found on Flasher a new one will be created.

A typical sequence using the #FOPEN command does look like as follows:

```
#FOPEN flasher.dat
#ACK
#OK
```

Note: Currently only one file can be open at the same time. If #FOPEN is send and another file is already open, Flasher will respond with:

```
#ACK
#ERR255:A file has already been opened
```

#FREAD <Offset>,<NumBytes>

The #FREAD command is used to read data from a file on Flasher. <Offset> specifies the offset in the file, at which data reading is started. <NumBytes> specifies the number of bytes which should be read.

A typical sequence when using the #FREAD command does look like as follows:

```
#FREAD 0,4
#ACK
#OK:04:466c6173
```

If the #FREAD command succeeds, Flasher will finally respond with a #OK:<NumBytes>:<Data> reply message. For more information about the Flasher reply messages, please refer to *Reply from Flasher* on page 68.

Note: In order to use the #FREAD command. A file has to be opened before, via the #FOPEN command. Otherwise Flasher will respond with the following sequence:

```
#ACK
#ERR255:No file opened
```

#FSIZE

The #FSIZE command is used to get the size of the currently opened file on Flasher.

A typical sequence when using the #FSIZE command does look like as follows:

```
#FSIZE
#ACK
#OK:10          // file on flasher which is currently open, has a size of 16 bytes
```

If the #FSIZE command succeeds, Flasher will respond with a #OK:<Size> reply message. For more information about the Flasher reply messages, please refer to *Reply from Flasher* on page 68.

Note: In order to use the #FREAD command. A file has to be opened before, via the #FOPEN command. Otherwise Flasher will respond with the following sequence:

```
#ACK
#ERR255:No file opened
```

#FWRITE <Offset>,<NumBytes>:<Data>

The #FWRITE command is used to write to a file on Flasher. <Offset> specifies the offset in the file, at which data writing is started. <NumBytes> specifies the number of bytes which are send with this command and which are written into the file on Flasher. <NumBytes> is limited to 512 bytes at once. This means, if you want to write e.g. 1024 bytes, you have to send the #FWRITE command twice, using an appropriate offset when sending it the second time.

<Offset> and <NumBytes> are expected in hexadecimal format.

```
#FWRITE 0,200:<Data>
#FWRITE 200,200:<Data>
```

The data is expected in hexadecimal format (two hexadecimal characters per byte). The following example illustrates the use of #FWRITE:

```
Data to be send: Hello !
ASCII values: 0x48, 0x65, 0x6C, 0x6C, 0x6F, 0x20, 0x21
```

```
#FWRITE 0,7:48656C6C6F2021
```

Note: In order to use the #FWRITE command a file has to be opened via the #FOPEN command, first. Otherwise Flasher will respond with the following sequence:

```
#ACK
#ERR255:No file opened
```

#FLIST

The #LIST command is used to list all files stored on the Flasher.

A typical sequence using the #FLIST command does look like as follows:

```
#FLIST
#ACK
FLASHER.INI           Size: 60
SERIAL.TXT            Size: 3
FLASHER.LOG           Size: 207
FOLDER                (DIR)
FOLDER\TEST1.CFG      Size: 2048
FOLDER\TEST1.DAT      Size: 12288
#OK
```

#MKDIR <Dirname>

The #MKDIR command is used to create a directory on Flasher. <Dirname> specifies the name of the new directory. <Dirname> may also specify a path to create a subdirectory.

A typical sequence using the #MKDIR command does look like as follows:

```
#MKDIR folder
#ACK
#OK
```

Note: If the directory can not be created because of a bad <Dirname> argument, Flasher will respond with:

```
#ACK
#ERR255:Failed to create directory
```

3.3.7 Reply from Flasher

The reply messages from Flasher 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 description 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 are defined:

#ACK

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

#NACK

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

#OK

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

#OK:<NumBytes>:<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).

#OK:<Size>

Flasher replies if #OK:<Size> if a #FSIZE command has been executed. <Size> is the size (in bytes) of the currently opened file. <Size> is send in hexadecimal format.

#STATUS:

Flasher 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.

Table 3.2: List of status messages that are currently defined

#ERRxxx

If any command other than #STATUS or #RESULT was terminated with an error, Flasher 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.
#ERR255	Undefined error occurred. This reply is followed by an error string.

Table 3.3: List of error code numbers which are currently defined

Chapter 4

Performance

The following chapter lists programming performance of common flash devices and microcontrollers.

4.1 Performance of MCUs with internal flash memory

4.1.1 Flasher ARM

The following table lists program and erase performance values of Flasher ARM for different controllers.

Microcontroller	Size [kByte]	Erase time [sec]	Program time [sec]	Verify time [sec]	Total time [sec]
Analog Devices	62	2.943	2.286	0.563	5.792
Atmel AT91SAM7S64	64	---	3.488	0.438	3.926
Atmel AT91SAM7S256	256	---	7.709	1.053	8.762
NXP LPC1768	512	3.740	8.559	5.092	17.391
NXP LPC2106	120	0.448	1.204	0.634	2.286
NXP LPC2129	248	0.449	2.916	1.347	4.712
NXP LPC2138	500	0.448	5.488	2.649	8.585
NXP LPC2148	500	0.448	5.632	2.721	8.801
NXP LPC2294	2048	0.808	15.976	9.669	26.453
NXP LPC2478	504	0.448	5.419	2.559	8.426
ST STM32F103ZE	512	0.028	18.763	3.939	22.730
ST STR711	272	0.429	5.476	4.742	10.647
ST STR912	544	1.167	12.907	5.236	19.310
TI TMS470R1B1M	1024	2.289	8.147	5.362	15.798

Table 4.1: List of performance values of MCUs with internal flash

4.1.2 Flasher RX

The following table lists program and erase performance values of Flasher RX.

Microcontroller	Size [kByte]	Erase time [sec]	Program time [sec]	Verify time [sec]	Total time [sec]
R5F56108	2.048	9.523	11.915	3.890	25.585

Table 4.2: List of performance values of MCUs with internal flash

4.1.3 Flasher PPC

The following table lists program and erase performance values of Flasher PPC.

Microcontroller	Size [kByte]	Erase time [sec]	Program time [sec]	Verify time [sec]	Total time [sec]
ST SPC560B50	576	4.747	4.159	1.929	10.917

Table 4.3: List of performance values of MCUs with internal flash

Chapter 5

Hardware

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

5.1 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.

VTref	1 •	• 2	Vsupply
nTRST	3 •	• 4	GND
TDI	5 •	• 6	GND
TMS	7 •	• 8	GND
TCK	9 •	• 10	GND
RTCK	11 •	• 12	GND
TDO	13 •	• 14	GND
RESET	15 •	• 16	GND
DBGREQ	17 •	• 18	GND
V5-Supply	19 •	• 20	GND

5.1.1 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 retimed, 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".
17	DBGREQ	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 DBGREQ 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.

Table 5.1: Flasher pinout JTAG

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.

5.1.2 Pinout SWD

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

VTref	1	•	•	2	Vsupply
Not used	3	•	•	4	GND
Not used	5	•	•	6	GND
SWDIO	7	•	•	8	GND
SWCLK	9	•	•	10	GND
Not used	11	•	•	12	GND
SWO	13	•	•	14	GND
RESET	15	•	•	16	GND
Not used	17	•	•	18	GND
V5-Supply	19	•	•	20	GND

The following table lists the J-Link / J-Trace 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 J-Link. 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 J-Link. 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 J-Link. 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 J-Link. This pin is not used by J-Link 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.)
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 J-Link.
19	5V-Target supply	Output	This pin is used to supply power to some eval boards. Not all J-Links supply power on this pin, only the KS (Kickstart) versions. Typically left open on target hardware.

Table 5.2: Flasher pinout SWD

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

5.1.3 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 J-Link commander. The following commands are available to control power:

Command	Explanation
<code>power on</code>	Switch target power on
<code>power off</code>	Switch target power off
<code>power on perm</code>	Set target power supply default to "on"
<code>power off perm</code>	Set target power supply default to "off"

Table 5.3: Command List

5.2 Flasher RX 14-pin connector

Flasher RX itself has a 20-pin JTAG connector mounted but comes with a 14-pin adapter for Renesas RX devices. This adapter also enables Flasher RX 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.

TCK	1 ● ● 2	GND
TRSTn	3 ● ● 4	EMLE
TDO	5 ● 6	---
---	7 ● 8	VTref
TMS	9 ● 10	---
TDI	11 ● ● 12	GND
nRES	13 ● ● 14	GND

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 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.
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 RX.
7	---	NC	This pin is not connected to Flasher RX.
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 RX.
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".

Table 5.4: Flasher RX pinout 14-pin connector

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

5.2.1 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 following commands are available to control power:

Command	Explanation
<code>power on</code>	Switch target power on
<code>power off</code>	Switch target power off
<code>power on perm</code>	Set target power supply default to "on"
<code>power off perm</code>	Set target power supply default to "off"

Table 5.5: Command List

5.3 Flasher PPC 14-pin connector

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

TDI	1 • • 2	GND
TDO	3 • • 4	GND
TCK	5 • • 6	GND
---	7 8	---
nRES	9 • • 10	TMS
VDDE7	11 • • 12	GND
nRDY	13 • • 14	JCOMP

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 PPC.
8	---	NC	This pin is not connected to Flasher PPC.
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 PPC (in order to enable the JTAG TAP controller on the target device).

Table 5.6: Flasher PPC pinout 14-pin connector

- All pins marked NC are not connected to Flasher PPC. Any signal can be applied here; Flasher PPC will simply ignore such a signal.
- Pins 2, 4, 6, 12 are GND pins connected to GND in Flasher PPC. They should also be connected to GND in the target system.

5.4 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 74
- *Pinout SWD* on page 75
- *Flasher RX 14-pin connector* on page 77
- *Flasher PPC 14-pin connector* on page 79

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

5.4.1 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.

5.4.2 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.

5.5 Adapters

5.5.1 JTAG Isolator

The JTAG Isolator can be connected between Flasher 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.

VCC	1	2	VCC
nTRST	3	4	GND
TDI	5	6	GND
TMS	7	8	GND
TCK	9	10	GND
RTCK	11	12	GND
TDO	13	14	GND
RESET	15	16	GND
N/C	17	18	GND
N/C	19	20	GND

5.5.1.1 Pinout

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

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 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 retimed, 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.

Table 5.7:

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

5.5.2 J-Link Needle Adapter

To connect to the J-Link OB 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.

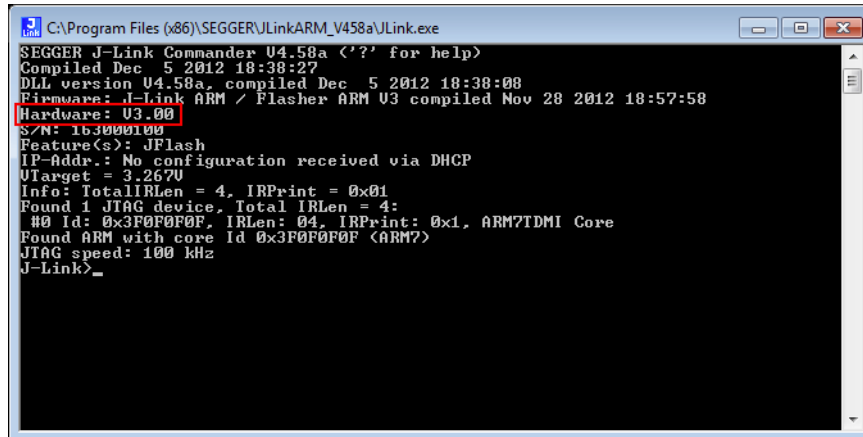


The J-Link Needle Adapter can be connected to J-Link via the *20-pin 0.1" JTAG* to a *10-pin needle connector*.

5.6 How to determine the hardware version

To determine the hardware version of your Flasher, the first step should be to look at the label at the bottom side of the unit. Flasher has the hardware version printed on the back label.

If this is not the case with your Flasher, you can use `JLink.exe` to determine your hardware version (if Flasher is in J-Link mode). As part of the initial message, the hardware version is displayed. For more information about how to ensure that Flasher is in J-Link mode, please refer to *J-Link mode* on page 26.



```

C:\Program Files (x86)\SEGGER\JLinkARM_V458a\JLink.exe
SEGGER J-Link Commander V4.58a <'?' for help>
Compiled Dec  5 2012 18:38:27
DLL version V4.58a, compiled Dec  5 2012 18:38:08
Firmware: J-Link ARM / Flasher ARM V3 compiled Nov 28 2012 18:57:58
Hardware: V3.00
S/N: 1630000100
Feature(s): JFlash
IP-Addr.: No configuration received via DHCP
UTarget = 3.2670
Info: TotalIRLen = 4, IRPrint = 0x01
Found 1 JTAG device, Total IRLen = 4:
#0 Id: 0x3F0F0F0F, IRLen: 04, IRPrint: 0x1, ARM7TDMI Core
Found ARM with core Id 0x3F0F0F0F <ARM7>
JTAG speed: 100 kHz
J-Link>_
  
```


Chapter 6

Support and FAQs

This chapter contains troubleshooting tips together with solutions for common problems which might occur when using Flasher. 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.

6.1 Contacting support

Before contacting support, make sure you tried to solve your problem by trying your Flasher 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 support@segger.com:

- A detailed description of the problem
- Flasher serial number
- Information about your target hardware (processor, board, etc.).
- `FLASHER.CFG`, `FLASHER.DAT`, `FLASHER.LOG`, `SERIAL.TXT` file from Flasher. To get these files, Flasher has to be in MSD mode. For more information about how to boot Flasher in MSD mode, please refer to *MSD mode* on page 32.

Flasher is sold directly by SEGGER.

6.2 Frequently Asked Questions

Maximum JTAG speed

- Q: What is the maximum JTAG speed supported by Flasher?
A: Flasher's maximum supported JTAG speed is 12MHz.

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.

Chapter 7

Background information

This chapter provides background information about flash programming in general. It also provides information about how to replace the firmware of Flasher manually.

7.1 Flash programming

Flasher comes with a DLL, which allows - amongst other functionalities - reading and writing RAM, CPU registers, starting and stopping the CPU, and setting breakpoints.

7.1.1 How does flash programming via Flasher work ?

This requires extra code. This extra code typically downloads a program into the RAM of the target system, which is able to erase and program the flash. This program is called RAM code and "knows" how to program the flash; it contains an implementation of the flash programming algorithm for the particular flash. Different flash chips have different programming algorithms; the programming algorithm also depends on other things, such as endianness of the target system and organization of the flash memory (for example 1 * 8 bits, 1 * 16 bits, 2 * 16 bits or 32 bits). The RAM code requires data to be programmed into the flash memory. The data is supplied by downloading it to RAM.

7.1.2 Data download to RAM

The data (or part of it) is downloaded to another part of the RAM of the target system. The Instruction pointer (PC) of the CPU is then set to the start address of the Ram code, the CPU is started, executing the RAM code. The RAM code, which contains the programming algorithm for the flash chip, copies the data into the flash chip. The CPU is stopped after this. This process may have to be repeated until the entire data is programmed into the flash.

7.1.3 Available options for flash programming

In general, there are two possibilities in order to use Flasher for flash programming:

- Using Flasher stand-alone to program the target flash memory (stand-alone mode)
- Using Flasher in combination with J-Flash to program the target flash memory (Flasher in "J-Link mode")

7.1.3.1 Using Flasher in stand-alone mode

In order to use the Flasher in stand-alone mode, it has to be configured first. For more information about how to setup Flasher for using in "stand-alone mode", please refer to *Setting up Flasher for stand-alone mode* on page 27.

7.1.3.2 J-Flash - Complete flash programming solution

J-Flash is a stand-alone Windows application, which can read / write data files and program the flash in almost any ARM system. For more information about J-Flash please refer to the *J-Flash User Guide*, which can be downloaded from our website <http://www.segger.com>.

Chapter 8

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 9

Literature and references

This chapter lists documents, which we think may be useful to gain a deeper understanding 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 (www.segger.com).
[J-Flash]	J-Flash User Guide	This document describes J-Flash. It is publicly available from SEGGER (www.segger.com).

Table 9.1: Literature and References

Index

A		N	
Adaptive clocking	92	nTRST	74, 93
C		O	
Cache cleaning	92	Open collector	93
Coprocessor	92	P	
D		Processor Core	93
Dirty data	92	R	
H		Remapping	93
Halfword	92	RESET	93
Host	92	RTOS	93
I		S	
ICache	92	Support	85, 91
ID	92	T	
IEEE 1149.1	92	TAP Controller	93
Image	92	Target	93
Instruction Register	92	TCK	74, 93
IR	92	TDI	74, 93
J		TDO	74, 93
J-Link		Test Access Port (TAP)	93
Adapters	81	Transistor-transistor logic (TTL)	94
Features	10	W	
Specifications	12	Word	94
Supported chips	56, 80		
Joint Test Action Group (JTAG)	92		
L			
Little-endian	92		
M			
Memory coherency	92		
Memory management unit (MMU)	93		
Memory Protection Unit (MPU)	93		

