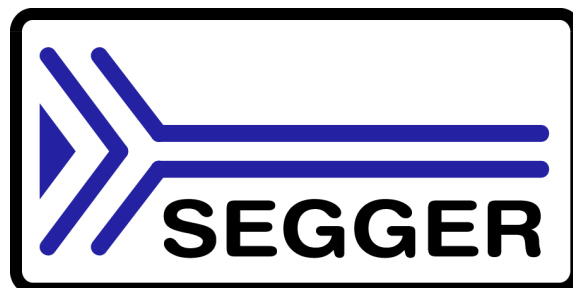# *J-Flash*

User guide of the stand-alone flash programming software

Software Version 5.10
Manual Rev. 0

Date: November 27, 2015

Document: UM08003

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.

© 2015 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

Manual versions

This manual describes the latest software version. If any error occurs, please inform us and we will try to assist you as soon as possible.

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

Manual version	Date	By	Explanation

5.02c Rev. 0	150914	RH	Chapter "Command Line Interface"
			 * Section "Command line options"
			   Added new commands:
			   -verifycrca, verifycrcs and verifycrcc

5.02a Rev. 0	150907	EL	Chapter "Command Line Interface" * Section "Command line options" updated.
5.00c Rev. 0	150611	EL	Chapter "Command Line Interface" * Section "Programming multiple targets in parallel" updated.
4.98 Rev. 2	150427	EL	Chapter "Command Line Interface" * Section "Command line options" Added new command: -ip and -USB

4.98 Rev. 1	150320	AG	Chapter "Background information"
			   Section "CRC of current data file"
			   polynomial corrected.

Table: List of manual versions

Manual version	Date	By	Explanation
4.98 Rev 0	150113	NG	Chapter "Command Line Interface" Changed "JFlashARM.exe" to "JFlash.exe".

4.96 Rev. 0	150109	EL	Chapter "Device specifics"
			  * Section "ST" updated.
			  * Section "Freescale" updated.

4.82 Rev. 0	140307	AG	Chapter "Device specifics" * Section "ST" updated.
4.80 Rev. 0	131220	AG	Chapter "Command Line Interface" * Section "Command line options" updated.
4.73c	130703	JL	Chapter "Getting Started" * Added Section "Start Dialog"

4.66 Rev. 1	130320	EL	Chapter "Settings"
			  * Section "CPU Settings"
			    Added description for the core ID "Mask" field

4.66 Rev. 0	130221	JL	Chapter "Introduction"
			  * Section "What is J-Flash"
			    Added Linux and Mac OSX

4.58 Rev. 0	121113	JL	Chapter "Command Line Interface"
			  * Section "Batch processing" updated.
			  * Section "Command line options" updated.

4.52 Rev. 0	120807	EL	Chapter "Getting Started"
			  * Section "Menu structure" updated
			Chapter "Settings"
			  * Section "CPU Settings" updated
			Chapter "Command Line Interface"
			  * Section "Programming multiple targets in
			    parallel" added.
			Chapter "Getting Started"
			  * Section "Sample Projects" updated.

4.51i Rev. 0	120724	EL	Chapter "Create a new J-Flash project"
			  * Section "Configuration for serial number
			    programming" added.

4.42b Rev. 0	120217	AG	Chapter "Background information" * Section "CRC of current data file" added.
4.24 Rev. 0	110216	AG	Chapter "Target systems" updated.
4.16 Rev. 1	100817	AG	Chapter "Command Line Interface" * Section "Command line options" corrected.
4.16 Rev. 0	100723	KN	Chapter "Settings" * Section "Init sequence" updated.
4.10 Rev. 4	091204	AG	Chapter "Device specifics" * Section "Freescale" added.
4.10 Rev. 2	090918	AG	Chapter "Command Line interface" * Section "Command line options" updated.
4.10 Rev. 1	090902	AG	Chapter "Device specifics" * Section "ST Microelectronics" updated.
4.10 Rev. 0	090825	AG	Chapter "Device specifics" * Section "ST Microelectronics" updated.
4.04 Rev. 1	090414	AG	Chapter "Introduction" * Section "What is J-Flash?" updated.

4.04 Rev. 0	090204	AG	Chapter "Command Line Interface"
			  * Section "Overview" updated.
			  * Section "Command Line Options" updated.

Table: List of manual versions

Manual version	Date	By	Explanation

3.97e Rev. 0	081204	KN	Chapter "Target systems"
			  * Section "Supported Flash Devices" updated
			Chapter "Settings"
			  * Section "Init sequence" corrected

3.91n Rev. 0	080923	AG	Chapter "Working with J-Flash" renamed to
			           "Create a new J-Flash project."
			Chapter "Create a new J-Flash project"
			Chapter "Settings"
			  * Section "Init sequence" updated.
			Chapter "Command Line Interface" updated.
			  * Section "Create a new J-FLash project"
			    updated.

3.90 Rev. 0	080811	AG	Chapter "Targets" * Section "Supported Microcontrollers" updated.

3.80 Rev. 2	080408	AG	Chapter "Licensing"
			  * Section "Introduction" added.
			  * Section "License types" added.

3.80 Rev. 1	080311	AG	Chapter "Target systems"
			  * Section "Supported Microcontrollers" updated.
			Chapter "Working with J-Flash"
			  * Section "Create a new J-Flash project"
			    updated.

3.80 Rev. 0	080206	SK	Chapter "Device specifics" added. Chapter "Target systems" * Section supported MCUs updated.
3.68 Rev. 1	070508	SK	Chapter "Installation" updated. Chapter "Command Line Interface": * Section "Batch processing" added. Various improvements.
3.66 Rev. 1	070322	SK	Chapter "Target systems" updated. Chapter "Getting started" updated.
3.46 Rev. 4	061222	SK	Section "About" and company description added.
3.46 Rev. 3	061124	OO	Chapter "Performance" updated.
3.46 Rev. 2	061121	OO	Chapter "Performance" updated.
3.46 Rev. 1	060929	TQ	Update supported target devices.
3.42 Rev. 1	060912	TQ	Update supported target devices.
3.36 Rev. 1	060801	TQ	Update supported target devices.
3.24 Rev. 1	060530	TQ	Update supported target devices.
3.00 Rev. 2	060116	OO	Screenshots updated.
3.00 Rev. 1	060112	TQ	Nothing changed. Just a new software version.
2.14	051025	TQ	Update supported target devices.
2.10	050926	TW	Added troubleshooting section.
2.04	050819	TQ	Nothing changed. Just a new software version.
2.02	050808	TW	Command line added.
2.00	050707	TW	Initial Version

Table: List of manual versions

Software versions

Refers to Release.html for information <u>about</u> the changes of the software versions.

About this document

Assumptions

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

- The software tools used for building your application (assembler, linker, C compiler)
- The C programming language
- The target processor
- DOS command line.

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

How to use this manual

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

Typographic conventions for syntax

This manual uses the following typographic conventions:

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

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 efficent 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 developes 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 1.1:

Table of Contents

Chapter 1

Introduction

The following chapter introduces J-Flash, highlights some of its features, and lists its requirements on host and target systems.

1.1   What is J-Flash?

J-Flash is a stand-alone flash programming software for PCs running Microsoft Windows. The following Microsoft Windows versions are supported:

*   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
*   Windows 8
*   Windows 8 x64

J-Flash has an intuitive user interface and makes programming flash devices convenient. J-Flash requires a J-Link, JTAG emulator for ARM cores, to interface to the hardware. It is able to program internal and external flash at very high speeds, upwards of 200 kB/sec depending on the chip. J-Flash has an approximate blank check speed of 16 MB/sec. Another notable feature is smart read back, which only transfers non-blank portions of the flash, increasing the speed of read back greatly. These features along with its ability to work with any ARM7 or ARM9 chip makes it a great solution for most projects.

1.1.1   Features

*   Any ARM7/ARM9, Cortex-M0/M1/M3/M4 and Renesas RX600 core supported
*   Microcontroller (internal flash) support.
*   Support for most external flash chips (For more information please refer to *Target systems* on page 75).
*   High speed programming: up to 200 KBytes/sec* (depending on flash device).
*   Very high speed blank check: approximately 16 MBytes/sec (depending on the chip).
*   Smart read back: only non-blank portions of flash are transferred and saved.
*   Free evaluation licenses available.
*   Verbose logging of all communication.
*   .hex, .mot, .srec, and .bin support.
*   Intuitive user interface.

* = Measured with J-Link ARM Rev.5 in DCC mode

1.2   Assumptions

This user manual assumes that you already possess working knowledge of the J-Link device. If you feel that your knowledge of J-Link is not sufficient, we recommend the J-Link manual, which describes the device and its use in detail.

1.3   Requirements

1.3.1   Host

J-Flash requires a PC running Microsoft Windows 2000 or Windows XP with a free USB port dedicated for a J-Link. A network connection is required only if you want to use J-Flash together with a remote J-Link server.

1.3.2   Target

A JTAG interface must be available on the target device to establish the connection with the host system. A network connection must be available if and only if it is desired to connect to the J-Link through the J-Link TCP/IP Server from a remote system.

Chapter 2

Licensing

The following chapter provides an overview of J-Flash related licensing options.

2.1   Introduction

J-Flash may be installed on as many host machines as you want. Without a license key you can still use J-Flash to open project files, read from connected devices, blank check target memory, verify data files and so on. However to actually program devices via J-Flash and J-link you are required to obtain a license key from us. A J-Flash license is bound to the serial number of a J-Link. Evaluation licenses which allow you to unlock the full potential of J-Flash for a limited period of time are available upon request. If you need an evaluation license key you only have to tell us the serial number of your J-Link which allows us to send you a proper key. In any case you need to have a license key for each J-Link you want to work with via J-Flash. The following sections describe common operations with reference to handling license keys.

2.2   License types

For J-Flash there are two different types of licenses which are explained below:

Built-in License

This type of license is easiest to use. The customer does not need to deal with a license key. The software automatically finds out that the connected J-Link contains the built-in license(s). This is the type of license you get if you order J-Link and the license at the same time, typically in a bundle.

Key-based license

This type of license is used if you already have a J-Link, but want to enhance its functionality by using J-Flash. In addition to that, the key-based license is used for trial licenses. To enable this type of license you need to obtain a license key from SEGGER. Free trial licenses are available upon request from *www.segger.com*. This license key has to be added to the J-Flash license management. How to enter a license key is described in detail in section *Key-based license* on page 18. Every license can be used on different PCs, but only with the J-Link the license is for. This means that if you want to use J-Flash with other J-Links, every J-Link needs a license.
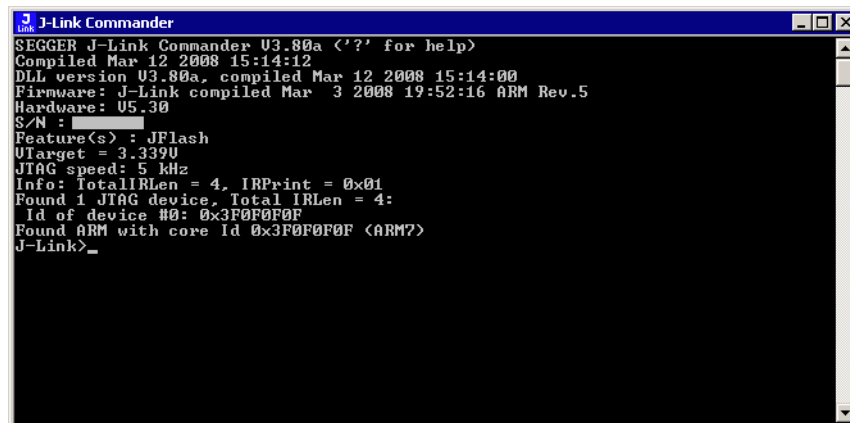
2.2.1   Built-in license

This type of license is easiest to use. The customer does not need to deal with a license key. The software automatically finds out that the connected J-Link contains the built-in license(s). To check what licenses the used J-Link have, simply open the J-Link commander (`JLink.exe`). The J-Link commander finds and lists all of the J-Link's licenses automatically, as can be seen in the screenshot below.

2.2.2   Key-based license

When using a key-based license, a license key is required in order to unlock the full potential of J-Flash. License keys can be added via the J-Flash license management. To get to the J-Flash license management just select Licenses... from the Help menu of the main window. Like the built-in license, the key-based license is only valid for one J-Link, so if another J-Link is used it needs a separate license.
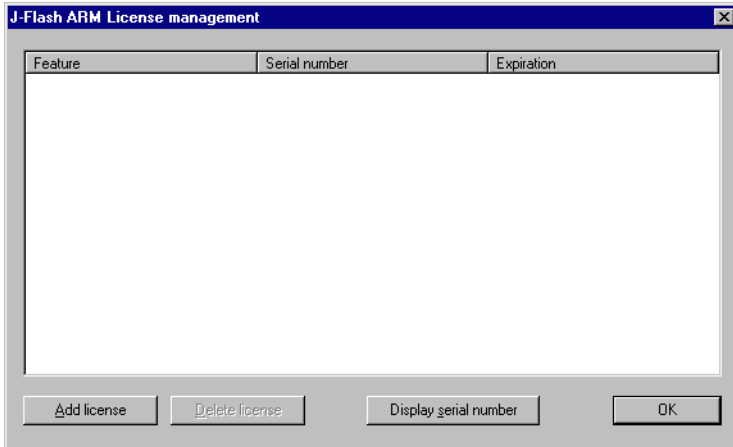
2.2.2.1   The serial number

The licensing dialog contains a button Display serial number. J-Flash tries to read the serial number of a connected J-Link if you press this button.
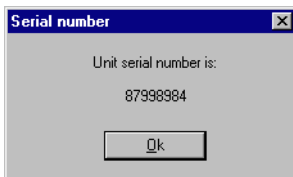
2.2.2.2   License management

The licensing dialog contains buttons to add and remove license keys. After you received a key from us, click on Add license to unlock J-Flash. Depending on the license you requested you are free to use J-Flash either for an unlimited or limited period of time. Enter the key into the Add license dialog and click OK to submit.
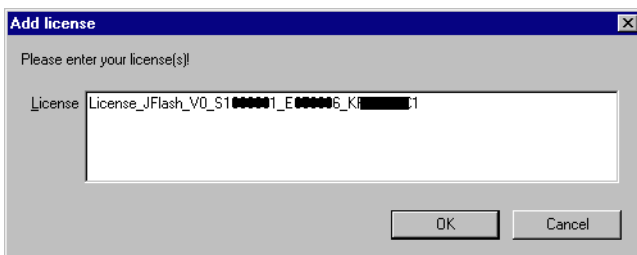
The licensing dialog will show the licenses together with their expiration date, the serial number they are bound to and the feature that is licensed by the respective key.
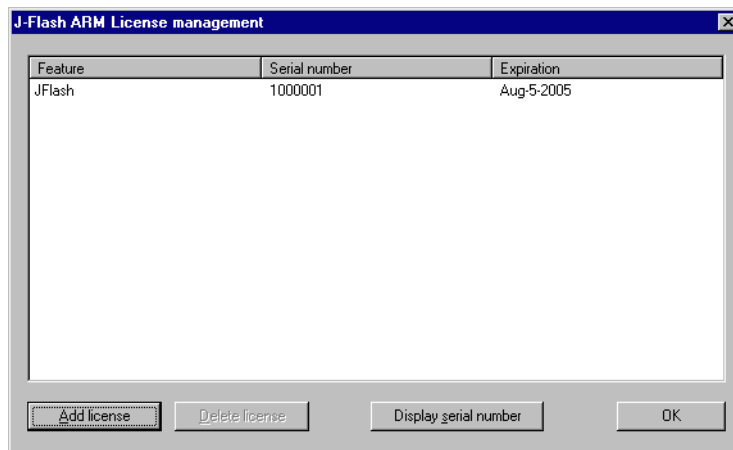
You may select individual license keys for removal. Click the Delete license button after selecting the key you want to remove. The key is deleted immediately without asking for confirmation and the licensed features become unavailable.

Chapter 3

Getting Started

This chapter presents an introduction to J-Flash. It provides an overview of the included sample projects and describes J-Flash's menu structure in detail.

3.1   Setup

The J-Link setup procedure required in order to work with the J-Flash is described in chapter 2 of the *J-Link / J-Trace User Guide*. The J-Link User Guide is part of the J-Link software package which is available for download under *www.segger.com*.

3.1.1   What is included?

The following table shows the contents of all subdirectories of the J-Link ARM software and documentation pack with regard to J-Flash:

Directory	Contents
.	The J-Flash application. Please refer to the J-Link manual for more information about the other J-Link related tools.
.\Doc	Contains the J-Flash documentation and the other J-Link related manuals.
.\ETC\JFlash\	Two *.csv files for the J-Flash internal management of supported MCU's und flash chips.
.\Sample\JFlash\Pro-jectFiles\	Contains sample projects with good default settings (see section *Sample Projects* on page 24 for further details).

Table 3.1: J-Flash directory structure

3.2   Using J-Flash for the first time

Start J-Flash from the Windows Start menu. J-Flash's main window will appear, which contains a log window at the bottom and the **Project window** of a default project on the left. The application log will initially display:

- The version and time of compilation for the J-Flash application.
- The version and time of compilation for the J-Link DLL.
- The number of supported flash devices.
- The number of supported MCU devices.
- The location of the default project.

A start dialog to choose a J-Flash project will pop up. It is recommanded to start from a template project. For further information about the start dialog, refor to the next chapter.

The Project window contains an overview of the current project settings (initially J-Flash opens a default project).
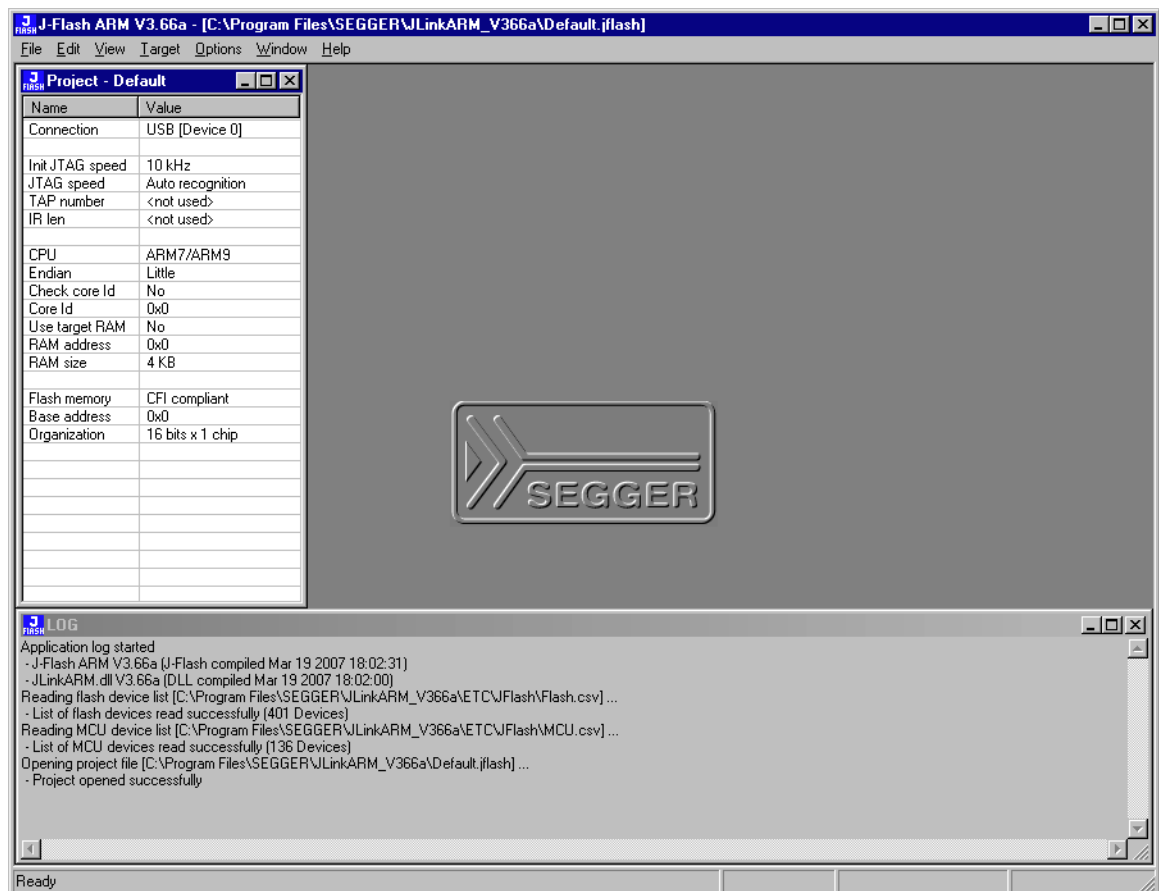
3.2.1   Sample Projects

If you are new to J-Flash, it might be a good idea to open one of our sample projects to familiarize yourself with the application. You find those project files in the Projects subdirectory of J-Flash's installation directory. Once you have opened a project file, the project window contains the relevant project settings, e.g. chip type, clock speed, RAM size etc. The settings are known to be good defaults for the respective devices. You may then continue to open your own data files to actually program your device. The table below contains the included project files together with a short description.

Project	Description
ADuC7020.jflash	Analog Devices ADuC7020 with internal flash memory
ADuC7030.jflash	Analog Devices ADuC7030 with internal flash memory
ADuC7032.jflash	Analog Devices ADuC7032 with internal flash memory
ADuC7038.jflash	Analog Devices ADuC7038 with internal flash memory
ADuC7039.jflash	Analog Devices ADuC7039 with internal flash memory
ADuC7060.jflash	Analog Devices ADuC7069 with internal flash memory
ADuC7124.jflash	Analog Devices ADuC7124 with internal flash memory
ADuC7229.jflash	Analog Devices ADuC7229 with internal flash memory
AM3505_CFI.jflash	Texas Instrument AM3505 with internal flash memory
AT91CAP9.jflash	Atmel AT91CAP9 with external flash memory
AT91FR40162.jflash	AT91FR40162 with internal AT49BV1614A flash memory
AT91M42800A.jflash	AT91M42800A with internal M29W200BB flash memory
AT91M55800A.jflash	AT91M55800 with Am29LV320DT flash memory
AT91R40008_AT91EB40A.jflash	AT91R40008 with external AT91EB40A flash memory
AT91RM9200_CSB337.jflash	Cogent CSB337 eval board with AT91RM9200
AT91RM9200_CSB637.jflash	Cogent CSB637 eval board with AT91RM9200
AT91RM9200_Digi_CC9U.jflash	Digi CC9U with AT91RM9200
AT91RM9200_EK.jflash	Atmel AT91RM9200 eval board
AT91SAM3S.jflash	Atmel AT91SAM3S3 with internal flash memory
AT91SAM3U4E.jflash	Atmel AT91SAM34UE with internal flash memory
AT91SAM7A1_EK.jflash	Atmel AT91SAM7A1 eval board with CFI compliant flash memory
AT91SAM7A3.jflash	Atmel AT91SAM7A3 with internal flash memory
AT91SAM7L128.jflash	Atmel AT91SAM7L128 with internal flash memory
AT91SAM7S32.jflash	AT91SAM7S-EK eval board with SAM7S32
AT91SAM7S64.jflash	AT91SAM7S-EK eval board with SAM7S64
AT91SAM7S128.jflash	AT91SAM7S-EK eval board with SAM7S128
AT91SAM7S256.jflash	AT91SAM7S-EK eval board with SAM7S256
AT91SAM7SE512.jflash	AT91SAM7SE-EK eval board with SAM7SE512
AT91SAM7X128.jflash	AT91SAM7X-EK eval board with SAM7X128
AT91SAM7X256.jflash	AT91SAM7X-EK eval board with SAM7X256

Table 3.2: List of sample J-Flash projects

Project	Description
`AT91SAM9260_CFI_1x16.jflash`	`AT91SAM9260` eval board with external CFI 1x16 flash memory
`AT91SAM9263_CSB737.jflash`	Cogent CSB337 eval board with `AT91SAM9263`
`AT91SAM9G20-EK_DataFlash.jflash`	`AT91SAM9G20`-EK eval board with `SAM9G20 data flash`
`AT91SAM9G45_CFI_2x8.jflash`	`AT91SAM9G45` eval board with external CFI 2x8 flash memory
`AT91SAM9XE512_AT91SAM9XE-EK.jflash`	`AT91SAM9XE`-EK eval board with `SAM9XE512`
`AyDeeKay_Chaveiro.jflash`	AyDeeKay Chaveiro with internal flash memory
`AyDeeKay_HeimdallSlave.jflash`	AyDeeKay HeimdallSlave with internal flash memory
`AyDeeKay_KamCho.jflash`	AyDeeKay KamCho with internal flash memory
`AyDeeKay_uSesame.jflash`	AyDeeKay uSesame with internal flash memory
`AyDeeKay_uSobek.jflash`	AyDeeKay uSobek with internal flash memory
`DA56KLF.jflash`	DSPGroup DA56KLF with internal flash memory
`DragonballMX1.jflash`	DragonballMX1 eval board with ST M29W400BB
`EFM32G890F128.jflash`	Energy Micro EFM32G890F128 with internal flash memory
`Ember_EM357.jflash`	Ember EM357 with internal flash memory
`Evaluator7T.jflash`	Evaluator7T eval board with SST39LF/VF400A flash memory
`GlynGraphicBaseBoard_TMPA900.jflash`	Glyn Graphic BaseBoard TMPA900 with external flash memory
`iMX27.jflash`	Freescale iMX27 with internal flash memory
`Itron_TRIFECTA.jflash`	Itron TRIFECTA with internal flash memory
`LH75411.jflash`	Sharp LH75411 with Macronix MX29LV320AB flash memory
`LH79520_LogicPD.jflash`	Sharp LH79520 with Intel 28F640J3 flash memory
`LH79524_LogicPD.jflash`	Sharp LH79524 with Sharp LH28F128SPHTD flash memory
`LH7A40x_LogicPD.jflash`	Sharp LH7A40x with Intel 28F640J3 flash memory (2 chips)
`LM3S811.jflash`	Luminary LMS811 with internal flash memory
`LM4F232H5.jflash`	Luminary LM4F232H5 with internal flash memory
`LPC1113.jflash`	NXP `LPC1113` with internal flash memory
`LPC11A_EEPROM.jflash`	NXP `LPC11A` with external EEPROM
`LPC1343.jflash`	NXP `LPC1343` with internal flash memory
`LPC1768.jflash`	NXP `LPC1768` with internal flash memory
`LPC1788_EmbeddedArtistsDevKit_CFI_1x16.jflash`	NXP `LPC1788` with external CFI 1x16 flash memory
`LPC1857.jflash`	NXP LPC2103 with internal flash memory
`LPC2103.jflash`	NXP LPC2103 with internal flash memory
`LPC2106.jflash`	NXP LPC2106 with internal flash memory
`LPC2129_MCB2100.jflash`	Keil MCB2100 eval board with NXP LPC2129
`LPC2138.jflash`	NXP LPC2138 with internal flash memory
`LPC2148.jflash`	NXP LPC2148 with internal flash memory
`LPC2210.jflash`	NXP LPC2210 with internal flash memory
`LPC2290.jflash`	NXP LPC2290 with internal flash memory
`LPC2294.jflash`	NXP LPC2294 with internal flash memory

Table 3.2: List of sample J-Flash projects

Project	Description
LPC2294_CFI_1x16.jflash	NXP LPC2294 with external 1x16 CFI flash memory
LPC2294_CFI_2x16.jflash	NXP LPC2294 with external 2x16 CFI flash memory
LPC2294_PhyCORE.jflash	NXP LPC2294 with external Am29DL800BT flash memory
LPC2366.jflash	NXP LPC2366 with internal flash memory
LPC2378.jflash	NXP LPC2378 with internal flash memory
LPC2478_1x16_CFI.jflash	NXP LPC2478 with external 1x16 CFI flash memory
LPC2478_EmbeddedArtists_1x16_CFI.jflash	Embedded Artists eval board with NXP LPC2478
LPC2888_NXP_Nohau.jflash	NXP Nohau eval board with NXP LPC2888
LPC2919_Hitex_LPC2919.jflash	Hitex eval board with LPC2919
LPC2919_MCB2900.jflash	Keil MCB2900 eval board with NXP LPC2919
LPC4088.jflash	NXP LPC4088 with internal flash memory
LPC2478_Olimex_LPC2478_STK.jflash	Olimex eval board with NXP LPC2378
MAC7111.jflash	Freescale MAC7111LC eval board with internal flash
MB86R03.jflash	Fujitsu MB86R03 with internal flash memory
MB9xFxxx.jflash	Fujitsu MB9xFxxx with internal flash memory
MC13224V_internalSPI.jflash	MC13224V with internal SPI flash
MK20DX128xxx5.jflash	OKI MK20DX128xxx5 with internal flash memory
MK21DN512xxx5.jflash	OKI MK21DN512xxx5 with internal flash memory
MK60N512.jflash	OKI MK60N512 with internal flash memory
MK70FN1M0.jflash	OKI MK70FN1M0 with internal flash memory
MKL25Z128xxx4.jflash	OKI MKL25Z128xxx4 with internal flash memory
ML67Q4002.jflash	OKI ML67Q4002 with internal flash memory
ML67Q4003.jflash	OKI ML67Q4003 with internal flash memory
ML67Q4050.jflash	OKI ML67Q4050 with internal flash memory
ML67Q4051.jflash	OKI ML67Q4051 with internal flash memory
ML67Q4060.jflash	OKI ML67Q4060 with internal flash memory
ML67Q4061.jflash	OKI ML67Q4061 with internal flash memory
NS7520_CC7U_352.jflash	Digi ConnectCore7U with NetSilicon NS7520 and external Fujitsu MBM29LV650U flash
NS7520_CC7U_355.jflash	Digi ConnectCore7U with NetSilicon NS7520 and external AMD Am29LV160BB flash
NS7520_CC7U_355_16bitSDRAM.jflash	NetSilicon NS7520 with external EN29LV800BT flash
NS9210_Digi_ConnectMe9210.jflash	NetSilicon NS9210 with external flash
NS9215_CC9P.jflash	NetSilicon NS9215 with external flash
NS9360.jflash	NetSilicon NS9360 with external AM29LV160DB flash (2 chips)
NS9750.jflash	NetSilicon NS9750 with Atmel AT49BV322A flash memory
PCF87750.jflash	NXP PCF87750 with internal flash memory
Q32M210.jflash	ON Semi Q32M210 with internal flash memory
RM48L9x.jflash	Texas Instruments RM48L9x with internal flash memory
RX600.jflash	Renesas R5F56108 with internal flash memory

Table 3.2: List of sample J-Flash projects

Project	Description
S3F441FX.jflash	Samsung S3F441FX with internal flash memory
S3F445HX.jflash	Samsung S3F445HX with internal flash memory
S3FN21x.jflash	Samsung S3FN21x with internal flash memory
S3FN41F.jflash	Samsung S3FN41F with internal flash memory
S3FN60D.jflash	Samsung S3FN60D with internal flash memory
SJA2010HL.jflash	NXP SJA2010 with internal flash memory
SJA2510HL.jflash	NXP SJA2510 with internal flash memory
SocLitePlus.jflash	NEC System-On-Chip Lite+ with internal memory
SPC560B50.jflash	ST SPC560B50 with internal flash memory
SPC560P50.jflash	ST SPC560P50 with internal flash memory
STM32F103RB.jflash	ST STM32F103RB with internal flash memory
STM32F103ZE_ST_MB672_CFI_1x16.jflash	ST STM32F103RB with external 1x16 flash memory
STM32F217ZG.jflash	ST STM32F217ZG with internal flash memory
STM32F303VC.jflash	ST STM32F303VC with internal flash memory
STM32F407IG.jflash	ST STM32F407IG with internal flash memory
STM32L152VB.jflash	ST STM32L152VB with internal flash memory
STM32L152VB_ProgUserOptionByte.jflash	ST STM32L152VB with internal flash memory
STM32L152ZD.jflash	ST STM32L152ZD with internal flash memory
STR710.jflash	ST STR710FZ2T6 with internal flash memory
STR711.jflash	ST STR711FR2T6 with internal flash memory
STR712.jflash	ST STR712FR2T6 with internal flash memory
STR730.jflash	ST STR730FZ2 with internal flash memory
STR750.jflash	ST STR750FV2 with internal flash memory
STR912.jflash	ST STR912FM44 with internal flash memory
STR912FAW47.jflash	ST STR912FAW47 with internal flash memory
STR912_Bootbank1.jflash	ST STR912 with internal flash memory
TMPA910.jflash	Toshiba TMPA910 with external flash memory
TMPM320_1x16_NOR.jflash	Toshiba TMPM320 with external NOR flash memory
TMPM330.jflash	Toshiba TMPM330 with internal flash memory
TMS470R1A64.jflash	TI TMS470R1A64 with internal flash memory
TMS470R1A128.jflash	TI TMS470R1A128 with internal flash memory
TMS470R1A256.jflash	TI TMS470R1A256 with internal flash memory
TMS470R1A288.jflash	TI TMS470R1A288 with internal flash memory
TMS470R1A384.jflash	TI TMS470R1A384 with internal flash memory
TMS470R1B1M.jflash	TI TMS470R1B1M with internal flash memory
TMS470R1B512.jflash	TI TMS470R1B512 with internal flash memory
TMS470R1VF689.jflash	TI TMS470R1VF689 with internal flash memory
TMS570LS20216.jflash	TI TMS570LS20216 with internal flash memory
TMS570LS20216_PLL_MCBTMS570.jflash	TI TMS570LS20216 with internal flash memory
TMS570LS3137.jflash	TI TMS570LS3137 with internal flash memory
Wavecom_WMP100_WMP100DevKit.jflash	Wavecome WMP100 external flash memory
XMC4500.jflash	Infineon XMC4500 with internal flash memory

**Table 3.2: List of sample J-Flash projects**

3.3   Start dialog

When starting J-Flash, by default a startup dialog pops up which gives the user vari-
ous options how to proceed.

For most users, the option "Create project from template" is recommended.

J-Flash comes with a large amount of sample/template projects which are pre-config-
ured for various targets, so in most cases the user just needs to select the appropri-
ate sample/template project for his target and can just start without further
configuration being necessary.

The options provided by the startup dialog are explained in more detail, below.
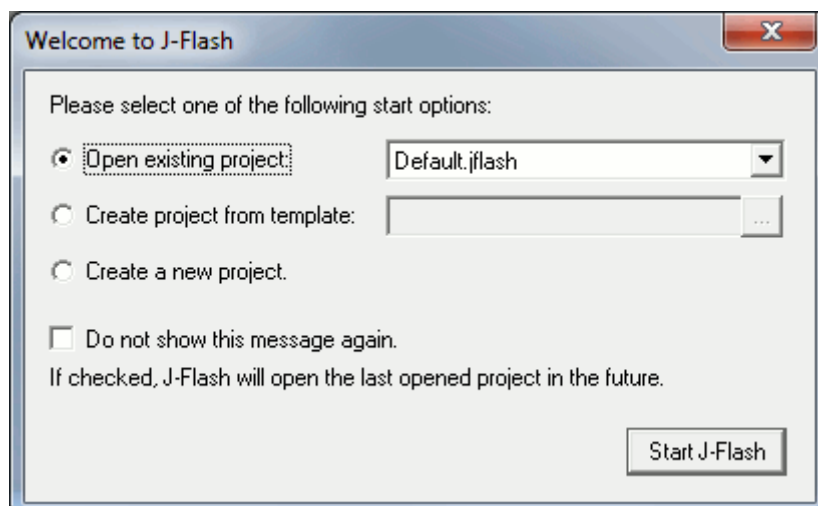
The startup dialog provides the following options:
*   **Open open existing project:** Select a project from the list of recent projects or
    select "Other..." to open another existing project.
*   **Create project from template**： Creates a new project based on an example
    project delivered with J-Flash. When selecting this and clicking the "..." button, a
    list of available example projects which are delivered with J-Flash is shown.
    As soon as an example project has been selected and "Start J-Flash" is clicked, J-
    Flash will ask where to save the new project. This makes sure that the pre-con-
    figured example projects delivered with J-Flash can not be accidentally overwrit-
    ten.
*   **Create new project:** Creates a blank/empty project with J-Flash default set-
    tings.

If "Do not show this message again." is checked, J-Flash will open the last
recently used project on future starts without showing the startup dialog again.

3.4   Menu structure

The main window of J-Flash contains seven drop-down menus (**File**, **Edit**, **View**, **Target**, **Options**, **Window**, **Help**). Any option within these drop-down menus that is followed by a three period ellipsis (...), is an option that requires more information before proceeding.

File menu elements

Command	Description
Open...	Opens a data file that may be used to flash the target device. The data file must be an Intel HEX file, a Motorola S file, or a Binary file (`.hex`, `.mot`, `.srec`, or `.bin`).
Merge	Merges two data files (`.hex`, `.mot`, `.srec`, or `.bin`). All gaps will be filled with FF. Find below a short example of merging two data files named, File0.bin and File1.bin into File3.bin. File0.bin --> Addr 0x0200 - 0x02FF File1.bin --> Addr 0x1000 - 0x13FF Merge File0.bin & File1.bin 0x0200 - 0x02FF Data of File0.bin 0x0300 - 0x0FFF gap (will be filled with 0xFF if image is saved as *.bin file) 0x1000 - 0x13FF Data of File1.bin Can be saved in new data file (File3.bin).
Save	Saves the data file that currently has focus.
Save As...	Saves the data file that currently has focus using the name and location given.
New Project	Creates a new project using the default settings.
Open Project...	Opens a J-Flash project file. Note that only one project file may be open at a time. Opening a project will close any other project currently open.
Save Project	Saves a J-Flash project file.
Save Project As...	Saves a J-Flash project file using the name and location given.
Close Project	Closes a J-Flash project file.
Export Setup File...	Exports a file that can be used to setup the J-Link. Please refer to the J-Link documentation for more information regarding J-Link setup files.
Recent Files >	Contains a list of the most recently open data files.
Recent Projects >	Contains a list of the most recently open project files.
Exit	Exits the J-Flash application.

Table 3.3: File menu elements

Edit menu elements

Command	Description
Relocate...	Relocates the start of the data file to the supplied hex offset from the current start location.
Delete range...	Deletes a range of values from the data file, starting and ending at given addresses. The End address must be greater than the Start address otherwise nothing will be done.
Eliminate blank areas...	Eliminates blank regions within the data file.

Table 3.4: Edit menu elements

View menu elements

Command	Description
Log	Opens and/or brings the log window to the active window.
Project	Opens and/or brings the project window to the active window.

Table 3.5: View menu elements

Target menu elements

Command	Description
Connect	Creates a connection through the J-Link using the configuration options set in the Project settings... of the Options drop-down menu.
Disconnect	Disconnects a current connection that has been made through the J-Link.
Show CFI info...	Reads the CFI query information of a CFI compliant flash device.
Test >	Two test functions are implemented "Generates test data" generates data which can be used to test if the flash can be programmed correctly. The size of the generated data file can be defined. "Tests up/download speed" writes data of an specified size to an defined address, reads the written data back and measures the up- and download speed.
Lock/Unlock sectors >	Sectors may be locked and unlocked. The soft lock and soft unlock work on a software only basis for those sectors that have been selected on the Flash tab of the Project Settings... found in the Options drop-down menu. If the software locks a sector with soft lock, it can easily be unlocked using the soft unlock feature. The hard lock and hard unlock work on a hardware only basis. If a sector is locked using the hard lock command, it can only be unlocked through hardware support. For example, some flash devices have a special PIN that must be set high or low to allow an unlock command.
Secure chip	Secures the MCU.
Unsecure chip	Unsecures the MCU.
Check blank	Checks flash to see if it is empty.
Fill with zero	Fills all selected flash sectors with zero. Some flash chips need this before erasing them.
Erase sectors	Erases all selected flash sectors.
Erase chip	Erases the entire chip.
Program	Programs the chip using the currently active data file.
Program & Verify	Programs the chip using the currently active data file and then verifies that it was written successfully.
Auto	The Auto command performs a sequence of steps. It connects to the device, erases sectors and programs the chip using the currently active data file before the written data is finally verified. The range of sectors to be erased can be configured through the Flash tab of the Project settings dialog and through the Global settings dialog. See chapter *Settings* on page 33 for further details.
Verify	Verifies the data found on the chip with the data file.

Table 3.6: Target menu elements

Command	Description
VerifyCRC >	Verifies the CRC. There are three ways in which the CRC can be verified. "Affected sectors" verifies the CRC of the affected sectors. "Selected sectors" verifies the CRC of the selected sectors. "Entire chip" verifies the CRC of the entire chip.
Read back >	Reads back the data found on the chip and creates a new data file to store this information. There are three ways in which the data can be read back. The Selected sectors identified on the Flash tab of the Project Settings... found in the Options drop-down menu may be read back. The Entire chip may be read back. A specified Range... may be read back.
Start Application	Starts the application found on the chip.

Table 3.6: Target menu elements

Options menu elements

Command	Description
Project settings...	Location of the project settings that are displayed in the snapshot view found in the Project window of the J-Flash application as well as various settings needed to locate the J-Link and pass specified commands needed for chip initialization.
Global settings...	Settings that influence the general operation of J-Flash.

Table 3.7: Options menu elements

Window menu elements

Command	Description
Cascade	Arranges all open windows, one above the other, with the active window at the top.
Tile Horizontal	Tiles the windows horizontally with the active window at the top.
Tile Vertical	Tiles the windows vertically with the active window at the left.

Table 3.8: Window menu elements

Help menu elements

Command	Description
J-Flash User's Guide	Shows this help file in a PDF viewer such as Adobe Reader.
J-Link User's Guide	Shows the J-Link User's Guide in a PDF viewer such as Adobe Reader.
Licenses...	Shows a dialog with licensing information. The serial number of a connected J-Link may be read and licenses added or removed.
About...	J-Flash and company information.

Table 3.9: Help menu elements

Chapter 4

Settings

The following chapter provides an overview of the program settings. Both, general and per project settings are considered.

4.1   Project Settings

Project settings are available from the Options menu in the main window or by using the ALT-F7 keyboard shortcut.

4.1.1   General Settings

This dialog is used to choose the connection to J-Link. The J-Link can either be connected directly over USB to the host system of J-Flash, or it can be connected through the J-Link TCP/IP Server running on a remote system. Refer to the J-Link manual for more information regarding the operation of J-Link and J-Link TCP/IP Server.

Since J-Flash version 3.74 can the complexity of user interface be selected. Select the **Engineering** checkbox if you want to setup your project or the **Simplified** checkbox if you use J-Flash in production environments. In the simplified user interface are some options disabled to decrease possible error sources in the production phase.

4.1.1.1 USB

If this option is checked, J-Flash will connect to J-Link over the USB port. You may change the device number if you want to connect more than one J-Link to your PC. The default device number is 0. For more information about how to use multiple J-Links on one PC, please see also the chapter "Working with J-Link" of the J-Link User's Guide.

4.1.1.2 TCP/IP

If this option is checked, J-Flash will connect to J-Link via J-Link TCP/IP Server. You have to specify the hostname of the remote system running the J-Link TCP/IP Server.

4.1.2   JTAG Settings

This dialog is used to configure the JTAG connection. You may change the JTAG speed or configure a JTAG scan chain with multiple devices.
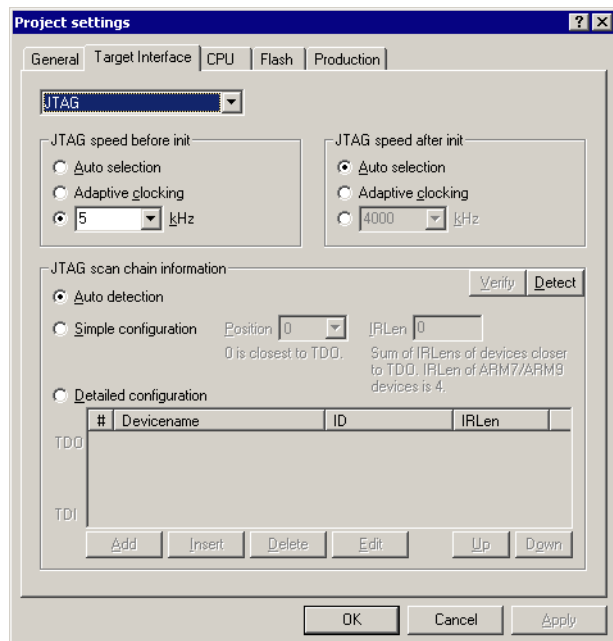

4.1.2.1   JTAG Speed

You can configure the JTAG speed used before and after initialization. The JTAG speed before init is used to communicate with the target before and during execution of the custom initialization sequence (described in section *CPU Settings* on page 37). The JTAG speed after init is used to communicate after executing the custom initialization sequence. This is useful if you have a target running at slow speed and you want to set up a PLL in the initialization sequence.

You can choose between automatic speed recognition, adaptive clocking or fixed JTAG speed. If you choose fixed JTAG speed you can select any value between 1kHz and 12MHz.

For more information about the different types of JTAG speed please see the chapter "Setup" of the J-Link User's Guide.

### 4.1.2.2 JTAG scan chain with multiple devices

This checkbox allows you to configure a JTAG scan chain with multiple devices on it. In a scan chain configuration with multiple devices, the TCK and TMS lines of all JTAG device are connected, while the TDI and TDO lines form a ring.
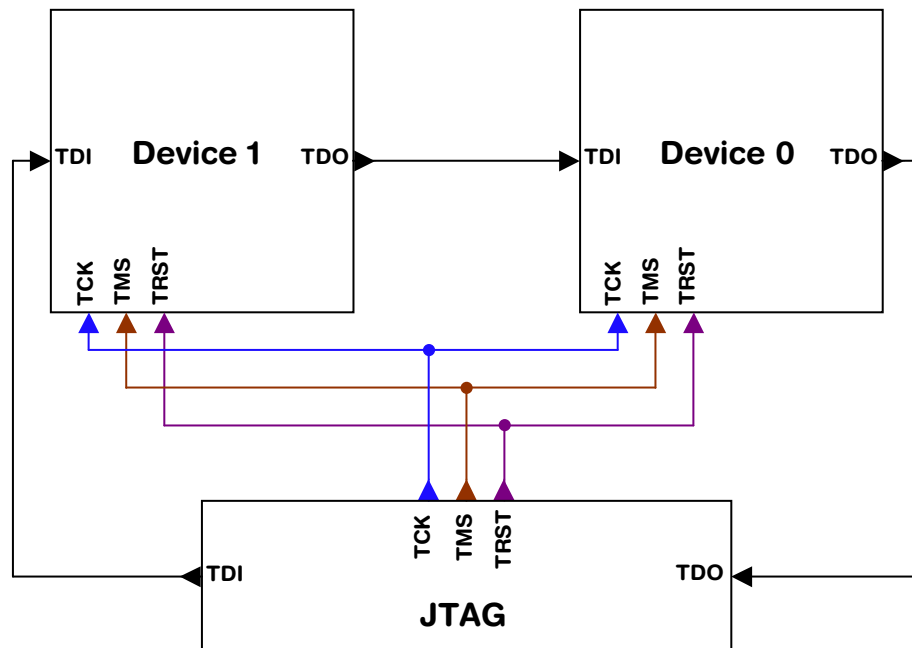
The position of the device to connect with J-Flash is selected from the Position drop-down menu. The Instruction Register length (IRLen) of a device is defined by its manufacturer. For ARM cores, the IRLen is always four, which is why the value of IRLen is by default set to four times the position indicated. This works fine for ARM only scan chains. However, if any non-ARM devices are introduced to the scan chain the IRLen must be modified accordingly.

4.1.3 CPU Settings

This dialog allows the selection of microcontroller dependent settings.
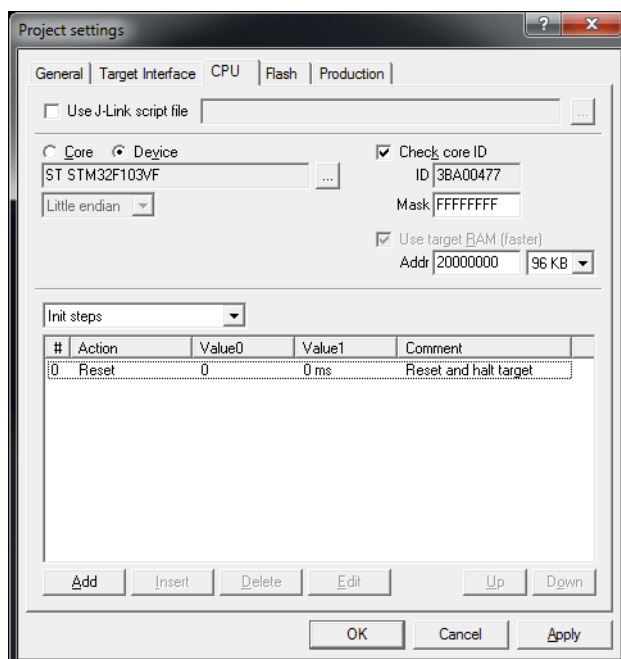
J-Flash can be used to program both external or internal flash memory. In order to use J-Flash with an external flash device, the proper **Core** must be selected.

To program internal flash devices choose the respective microcontroller in the **Device** list. If your microcontroller is not found on this list, contact SEGGER as new microcontrollers are continuously being added.

4.1.3.1 Core

Select **Generic ARM7/ARM9** or **XSCALE** depend on the used MCU architecture version to program external flash.

4.1.3.2 Device

Select the respective microcontroller from the list to program internal flash devices.

4.1.3.3 Clock

The correct clock frequency in Hz of your MCU is required to guarantee accurate operation of J-Flash. J-Flash uses default the Auto detection feature. We recommend to change this default behavior only if you know what you do. If you deactivate the Auto detection feature take into account, that you have to modify the value in this dialog if you set up a PLL or otherwise change the clock frequency in the init sequence.

4.1.3.4 Endianess

The compatible endianness of the selected device is set automatically. The endianess must be only explicit defined, if you select the **Core** family to program external flash. Select **Little endian** or **Big endian** from drop-down menu accordant to your core.

4.1.3.5 Check core ID

If the core ID is known for the device to be programmed, it can be used to verify that the device in communication via the J-Link is the intended device. The core ID for all listed devices is known, so is this option selected automatically if you select a device from the **Device** drop-down menu and can not be modified. If you select the core family from the **Core** drop-down menu, you can modify the default core ID.

Mask

This option allows the user to mask out specified bits of the core ID. All bits set to 0 in "Mask" are not taken into account when comparing the Code ID found by J-Link with the Core ID entered in J-Flash.

Example:

Values	Check result
Core ID entered: 0x3BA00477 Core ID found: 0x4BA00477 Mask: 0xFFFFFFFF	Failed
Core ID entered: 0x3BA00477 Core ID found: 0x4BA00477 Mask: 0x0FFFFFFF	Passed

The code ID check works as follows:

```
CoreIDFound &= Mask;
CoreIDEntered &= Mask;
if (CoreIDFound != CoreIDEntered) {
  return Error;                       // Core ID check failed.
}
```

4.1.3.6   Use target RAM

You may enable the use of target RAM to speed up flash operations. To use the target RAM, a start location in RAM and the amount of RAM to be used must be entered.

4.1.3.7   Init steps

Many microcontrollers require an initialization sequence for different reasons: When powered on, the PLL may not be initialized, which means the chip is very slow or a watchdog must be disabled manually. To use these chips you must first perform the required initialization.

This dialog allows the user to enter a custom initialization sequence using a pre-defined list of operations. After choosing an operation and corresponding values to be associated with the operation, a comment may be added to make it easier for others to determine its effect. The following list shows all valid commands which can be used in an init sequence:

Command	Value0	Value1	Description
Delay	--	Length of the delay	Sets a delay.
DisableMMU	--	--	Disables the MMU.
Disable Checks	--	--	Disables JTAG checks. Some CPUs (e.g. TMS470R1B1M) report JTAG communication errors while initializing, so that they can not be programmed if the JTAG communication checks are enabled.
Enable Checks	--	--	Enables JTAG checks. This option is activated by default.
Go	--	--	Starts the CPU
Halt	--	--	Halts the CPU
Reset	J-Link reset type	Length of the delay	Resets the CPU. Refer to the J-Link / J-Trace user guide for an detailed explanation of the different reset types.
Read 8bit	Address (Hex)	--	Reads 8bit from a given address and stores the value in an internal variable.
Read 16bit	Address (Hex)	--	Reads 16bit from a given address and stores the value in an internal variable.
Read 32bit	Address (Hex)	--	Reads 32bit from a given address and stores the value in an internal variable.
SetAllowRemo-teRead	--	On/Off	This option defines if the emulator (remote) or the host handles the read access to target. This option is activated by default to enhance the performance.
SetAllowRemote-Write	--	On/Off	This option defines if the emulator (remote) or the host handles the write access to target. This option is activated by default to enhance the performance.

Table 4.1: J-Flash init sequence commands

Command	Value0	Value1	Description
Verify 8bit	Address (Hex)	Data	Verifies whether 8bit data on a declared address is identical to the declared 8bit data.
Verify 16bit	Address (Hex)	Data (Hex)	Verifies whether 16bit data on a declared address is identical to the declared 16bit data.
Verify 32bit	Address (Hex)	Data (Hex)	Verifies whether 32bit data on a declared address is identical to the declared 32bit data.
Write 8bit	Address (Hex)	Data (Hex)	Writes 8bit data to a given address.
Write 16bit	Address (Hex)	Data (Hex)	Writes 16bit data to a given address.
Write 32bit	Address (Hex)	Data (Hex)	Writes 32bit data to a given address.
Write Register	Register	Value	Writes data into a register.
Write JTAG IR	Command	--	Writes a command in the JTAG instruction register.
Write JTAG DR	NumBits	Data (Hex)	Writes a declared number of bits into the JTAG data register.
Var AND	--	Value (Hex)	Logical AND combination of the internal variable with a given value.
Var OR	--	Value (Hex)	Logical OR combination of the internal variable with a given value.
VAR XOR	--	Value (Hex)	Logical XOR combination of the internal variable with a given value.
VAR BEQ	Index	--	Checks if the internal variable is equal to 0. Performs jump to index on match.
VAR BNE	Index	--	Checks if the internal variable is not equal to 0. Performs jump to index on match.
Var Write 8bit	Address (Hex)	Data (Hex)	Writes 8bit data of the internal variable to a given address.
Var Write 16bit	Address (Hex)	Data (Hex)	Writes 16bit data of the internal variable to a given address.
Var Write 32bit	Address (Hex)	Data (Hex)	Writes 32bit data of the internal variable to a given address.

Table 4.1: J-Flash init sequence commands

4.1.3.8   Exit steps

Those steps will be performed immediately after the target has been successfully programmed. In case of verify is checked in the production settings (**Options** -> **Project settings...** -> **Production settings tab**), those steps will be performed after verify.

The Exit steps can be used to do some special handling after programming, for example to set some security bits in order to secure the chip.

Note: Exit steps are only performed for "Target->Auto" operations.

4.1.4   Flash Settings

This dialog is used to select and configure the flash device to operate with. The listed options of the Flash settings menu are dependent on the selection in the **CPU** settings dialog. If you have selected a core family to program external flash memory, the menu should look similar to the screenshot below.
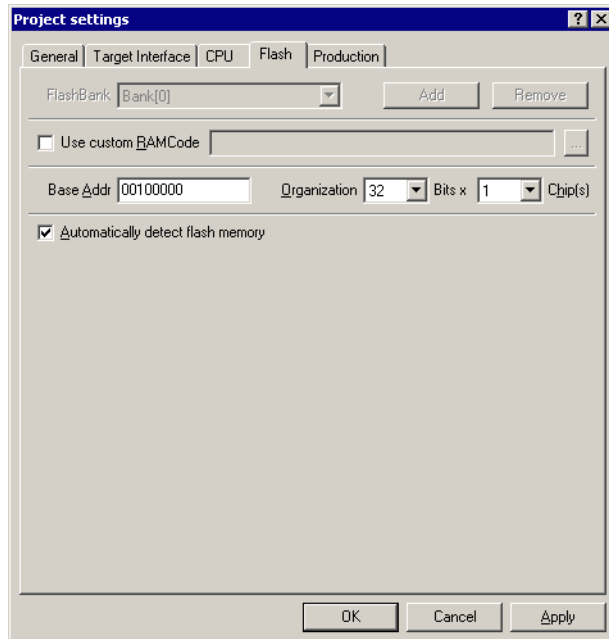
If you have selected a specific device to program the flash of these device, the menu should look similar to the screenshot below.
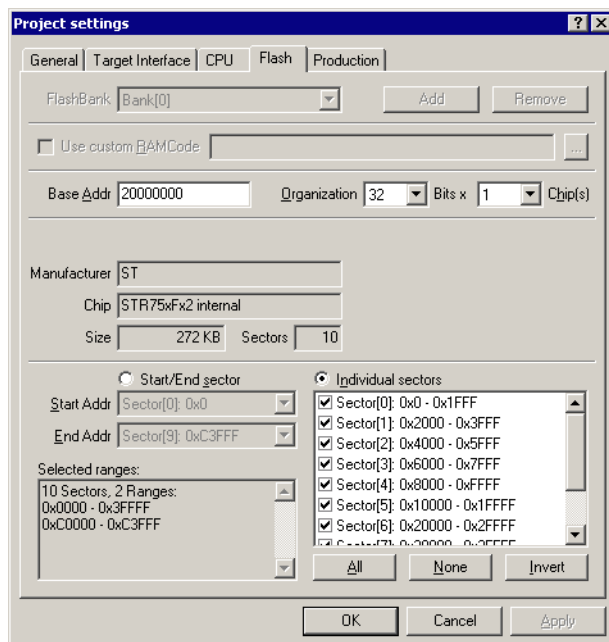
4.1.4.1   Base Address

You may enter the base address of the selected flash memory. The default value is 0.

4.1.4.2   Organization

You should select the buswidth and the number of flash chips connected to the address and data bus of the MCU.

4.1.4.3   Select flash device

You can select a device manually or use the J-Flash **Auto Detection** feature. The auto detection feature is select by default. It supports both CFI compliant flash memory chips and non CFI compliant chips. You can select a device manually, if you deselect the Auto Detection checkbox and click on the **Select flash device** button.

After invoking this button a table will be presented. The table may be filtered using the manufacturer name. The chip and its attributes (manufacturer name, device name, size, number of sectors, eight bit identifier, sixteen bit identifier, bus width) must be selected from this table. If the flash chip is not found please contact SEGGER, as devices are continuously being added to this list.
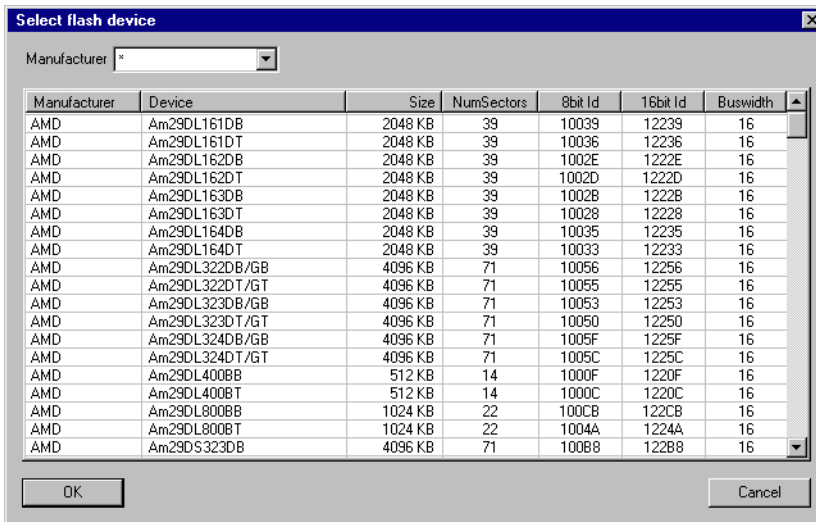

4.1.4.4   ID checking

There are two other check boxes that are of interest in this subsection which are "Check manufacturer flash Id" and "Check product flash Id". These check boxes should be selected to confirm the type of device that is in communication with J-Flash.

4.1.4.5   Sector selection

The final section of this dialog indicates the sectors to be acted upon, whether they are to be cleared, read back, or written. An individual or series of sectors may be selected from the predetermined valid range.

4.1.5   Production settings
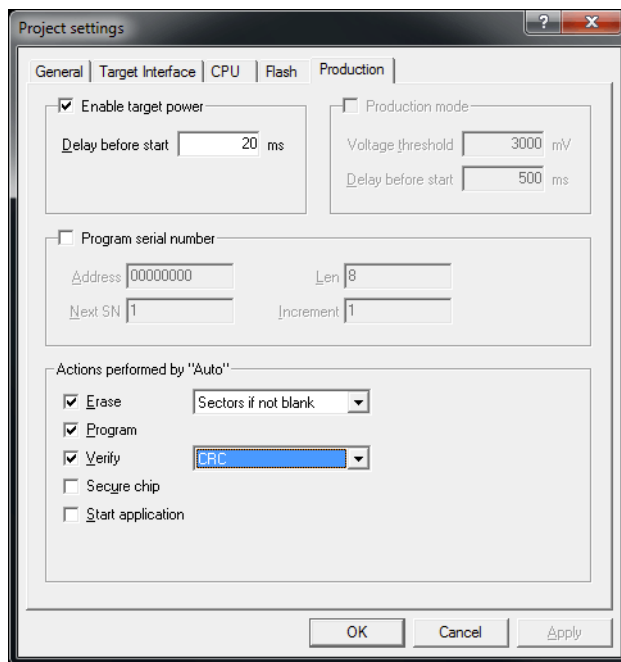
Enable target power

Enables 5V target power supply via pin 19 of the emulator.

Can be used for targets which can be powered through the emulator for production.

`Delay before start` defines the delay (in ms) after enabling the target power supply and before starting to communicate with the target.

Program serial number

J-Flash supports programming of serial numbers into the target in two ways. For a detailed description how to use the serial number programming feature please refer to *Serial number programming* on page 59.

Actions performed by "Auto"

The checked options will be performed when auto programming a target (**Target** -> **Auto**, shortcut: F7). The default behaviour is **Erase sectors if not blank**, **Program** and **Verify CRC**. You can optional include **Secure chip** and **Start application**. Find below a table which describes the commands:

| Command	Description
Erase	Performs an erase depending on the settings, selected in the drop down box. • `Sectors`: Erases all sectors which are effected by the image to be programmed. • `Sectors if not blank`: Erases all sectors which are both, effected by the image to be programmed and not already blank. • `Chip`: Erase the entire chip independent of the content.
Program	Programs the data file.

Table 4.2: Actions performed by "Auto"

Command	Description
Verify	Verifies the program data. • CRC: Verifies data via a high optimized CRC calculation (recommended verification method). • Complete data: Verifies data by reading it back.
Secure chip	Secures the device if supported by algorithm
Start application	Starts application after programming/verify completed. Needs reset bin to be conccted to J-Link.

Table 4.2: Actions performed by "Auto"

4.2   Global Settings

Global settings are available from the Options menu in the main window.
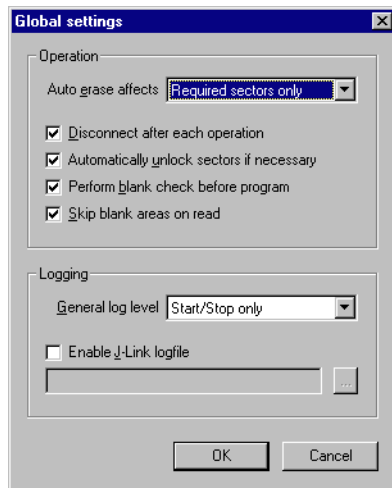
4.2.1   Operation

You may define the behavior of some operations such as "Auto" or "Program & Verify".

4.2.1.1   Disconnect after each operation

If this option is checked, connection to the target will be closed at the end of each operation.

4.2.1.2   Automatically unlock sectors

If this option is checked, all sectors affected by an erase or program operation will be automatically unlocked if necessary.

4.2.1.3   Perform blank check

If this option is checked, a blank check is performed before any program operation to check if the affected flash sectors are completely empty. The user will be asked to erase the affected sectors if they are not empty.

4.2.1.4   Skip blank areas on read

If this option is checked, a blank check is performed before any read back operation to check which flash areas need to be read back from target. This improves performance of read back operations since it minimizes the amount of data to be transferred via JTAG and USB.

4.2.2   Logging

You may set some logging options to customize the log output of J-Flash.

4.2.2.1   General log level

This specifies the log level of J-Flash. Increasing log levels result in more information logged in the log window.

4.2.2.2   Enable J-Link logfile

If this option is checked, you can specify a file name of the J-Link logfile. The J-Link logfile differs from the log window output of J-Flash. It does not log J-Flash operations performed. Instead of that, it logs the J-Link ARM DLL API functions called from within J-Flash.

Chapter 5

Command Line Interface

This chapter describes the J-Flash command line interface. The command line allows using J-Flash in batch processing mode and other advanced uses.

5.1   Overview

In addition to its traditional Windows graphical user interface (GUI), J-Flash supports a command line mode as well. This makes it possible to use J-Flash for batch processing purposes. All important options accessible from the menus are available in command line mode as well. If you provide command line options, J-Flash will still start its GUI, but processing will start immediately.

The screenshot below shows the command line help dialog, which is displayed if you start J-Flash in a console window with `JFlash.exe -help` or `JFlash.exe -?`
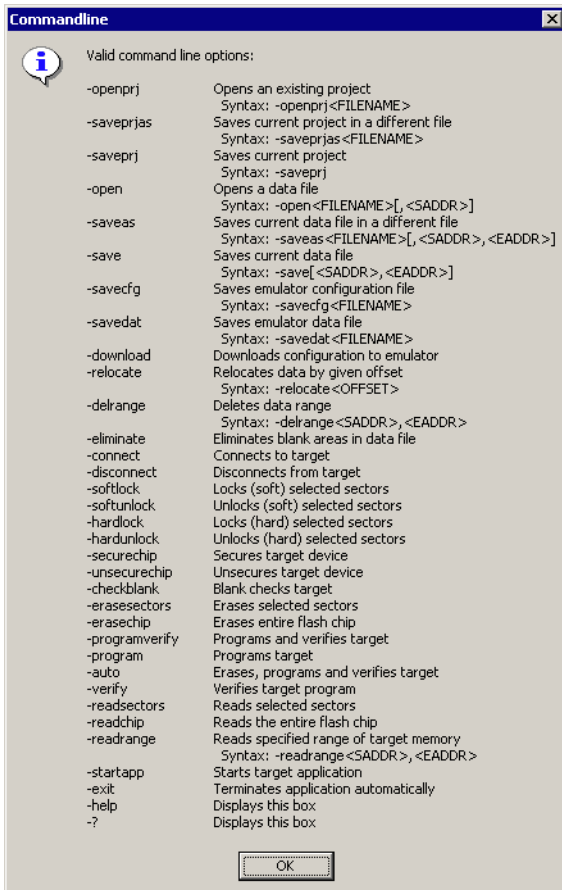
5.2 Command line options

This section lists and describes all available command line options. Some options accept additional parameters which are enclosed in angle brackets, e.g. <FILE-NAME>. If these parameters are optional they are enclosed in square brackets too, e.g. [<SADDR>]. Neither the angel nor the square brackets must be typed on the command line, they are used here only to denote (optional) parameters. Also, note that a parameter must follow immediately after the option,
e.g. `JFlash.exe -openprjC:\Projects\Default.jflash`.

The command line options are evaluated in the order as they are passed to J-Flash, so please ensure that a project + data file have already been opened when evaluating a command line option which requires this.

It is recommended to always use `-open<FILENAME>[,<SADDR>]` to make sure the right data file is opened.

All command line options return 0 if the processing was successfully. An return value unequal 0 means that an error occured.

Option	Description
-?	Display help dialog.
-auto	Erase, program and verify target.
-checkblank	Blank check target.
-connect	Connect to target.
-delrange<SADDR>,<EADDR>	Delete data in the given range.
-disconnect	Disconnect from target.
-download	Downloads configuration to emulator.
-eliminate	Eliminate blank areas in data file.
-erasechip	Erase the entire flash chip.
-erasesectors	Erase selected sectors.
-exit	Exit J-Flash.
-hardlock	Locks (hard) selected sectors.
-hardunlock	Unlocks (hard) selected sectors.
-help	Display help dialog.
-merge<FILENAME> -merge<FILENAME>.bin,<ADDR>	Merges a given file with the one currently opened in J-Flash. Note that when passing a .bin file via this command, also the start-address where it shall be merged into the file currently openend in J-Flash must be given, since .bin files do not contain any address information.
-open<FILENAME>[,<SADDR>]	Open a data file. Please note that the <SADDR> parameter applies only if the data file is a *.bin file.
-openprj<FILENAME>	Open an existing project file. This will also automatically open the data file that has been recently used with this project.
-program	Program target.
-programverify	Program and verify target.
-readchip	Read entire flash chip.
-readsectors	Read selected sectors.
-readrange<SADDR>,<EADDR>	Read specified range of target memory.
-relocate<OFFSET>	Relocate data by the given offset.
-save[<SADDR>,<EADDR>]	Save the current data file. Please note that the parameters <SADDR>,<EADDR> apply only if the data file is a *.bin file or *.c file.

Table 5.1: J-Flash command line options

Option	Description
-saveas<FILE-NAME>[,<SADDR>,<EADDR>]	Save the current data file into the specified file. Please note that the parameters <SADDR>, <EADDR> apply only if the data file is a *.bin file or *.c file.
-savecfg<FILENAME>	Saves emulator config file.
-savedat<FILENAME>	Saves emulator data file.
-saveprj	Save the current project.
-saveprjas<FILENAME>	Save the current project in the specified file.
-securechip	Secures target device.
-setrxidcode<IDCodeString>	Sets the RXIDCode to be used by the DLL. Syntax: -setrxidcode<IDCodeString> (32 characters, 16 IDCode bytes)
-softlock	Lock (soft) selected sectors.
-softunlock	Unlock (soft) selected sectors.
-startapp	Start target application.
-unsecurechip	Unsecures target device.
-verify	Verify target memory.
-verifycrca	Verify CRC of affected sectors
-verifycrcs	Verify CRC of selected sectors
-verifycrcc	Verify CRC of entire flash chip
-usb<SN>	Overrides connection settings to USB S/N.
-ip<xxx.xxx.xxx.xxx> -ip<HostName>	Overrides connection settings to IP S/N.

Table 5.1: J-Flash command line options

5.3   Batch processing

J-Flash can be used for batch processing purposes. All important options are available in command line mode as well. If you provide command line options, J-Flash will still start its GUI, but processing will start immediately.

The example batchfile below will cause J-Flash to perform the following operations:

1. Open project C:\Projects\Default.jflash
2. Open bin file C:\Data\data.bin and set start address to 0x100000
3. Perform "Auto" operation in J-Flash (by default this performs erase, program, verify)
4. Close J-Flash

The return value will be checked and in case of an error an error message displayed. Adapt the example according to the requirements of your project.

```
@ECHO OFF

ECHO Open a project and data file, start auto processing and exit
JFlash.exe -openprjC:\Projects\Default.jflash -openC:\Data\data.bin,0x100000 -auto -
exit
IF ERRORLEVEL 1 goto ERROR

goto END

:ERROR
ECHO J-Flash ARM:   Error!
pause

:END
```

Starting J-Flash minimzed

Adapt this example call to start J-Flash minimized:

```
start /min /wait "J-Flash" "JFlash.exe" -openprjC:\Projects\Default.jflash          \
-openC:\Data\data.bin,0x100000 -auto -exit
```

Note, that every call of JFlash.exe has to be completed with the -exit option, otherwise the execution of the batch file stops and the following commands will not be processed.

5.4   Programming multiple targets in parallel

In order to program multiple targets in parallel using J-Flash, the following is needed:

- J-Link / Flasher needs to be configured to allow to connect multiple ones to one PC at the same time. Please refer to *"UM08001 Working with J-Link and J-Trace" -> "Connecting multiple J-Links / J-Traces to your PC"*
- One J-Flash project (containing the configuration).

Basically, J-Flash connects to a specific J-Link, configured in the project settings but there is a command line option available, which allows to temporary override this setting. Therefore, only one J-Flash project is needed.

Find below a small sample which shows how to program multiple targets in parallel.

```
@ECHO OFF
setlocal
set "lock=%temp%\wait%random%.lock"

ECHO Launches 3 processes in parallel, where each process calls J-Flash with
different serial number as connection method. The stream of each process is
redirected to a lock file so we can wait until each process
has been completed.
start "" 9>"%lock%1" StartJFlash.bat 164000000
start "" 9>"%lock%2" StartJFlash.bat 164000001
start "" 9>"%lock%3" StartJFlash.bat 164000002
REM Wait until lock files are released and delete them afterwards
1>nul 2>nul ping /n 3 ::1
for %%N in (1 2 3) do (
  (call ) 9>"%lock%%%N" || goto :Wait
) 2>nul
del "%lock%*"
pause
```

StartFlash.bat:

```
@ECHO OFFECHO Open a project and data file and exit
start /wait "J-Flash" "JFlash.exe" -usb%1 -connect -exit
IF ERRORLEVEL 1 goto ERROR
goto END
:ERROR
ECHO %ERRORLEVEL%
ECHO J-Flash: Error! SN: %1
pause
exit
:END
ECHO J-Flash: Succeed!
exit
```

Note: Each call of `JFlash.exe` has to be completed with the `-exit` option, otherwise the execution of the batch file is and the following commands will not be processed.

Chapter 6

Create a new J-Flash project

This chapter contains information about the required steps how to setup a new J-Flash project.

6.1   Creating a new J-Flash project

Before creating a new J-Flash project, you should have an understanding of your target system:

- Take a look at the schematic and the documentation of your CPU / SOC.
- Make sure the CPU runs at a decent speed (at least a few MHz, not just 32kHz)
- If necessary, enable & select a PLL as clock source.
- Locate RAM (Ideally on-chip RAM, even if it is just a 4KB) in the chip documentation.
- If necessary, use external RAM (usually SDRAM). You may have to setup the external bus interface.
- If necessary, use external FLASH. You may also have to setup the external bus interface to program the external flash since in a lot of cases, it allows per default just reading of flash memory.
- Last but not least, you should make sure the JTAG speed is as high as possible (on ARM-S cores with RTCK, Adaptive is usually a good choice; if not, 8Mhz or 12MHz is ideal. However, this should be the last step)

Note: Initialization of the PLL and the external bus interface has to be done in the init sequence of the project.

In the following all the necessary steps to create a project file, are explained.

1. Select **File** -> **New Project** to open a new project.
2. Open the **Project Settings** context menu. Select **Options** -> **Project Settings** or press ALT-F7 to open the **Project settings** dialog and select the type of connection to J-Link.


Select **Engineering (More options, typically used for setup)**.

3. Define the **JTAG speed before init** and the **JTAG speed after init**. The default settings work without any problem for the most targets.
    Since software version 3.80 J-Flash supports SWD. To select SWD as target inter-face, simply select **SWD** from the dropdown box and define the **SWD speed before init** and the **SWD speed after init.**
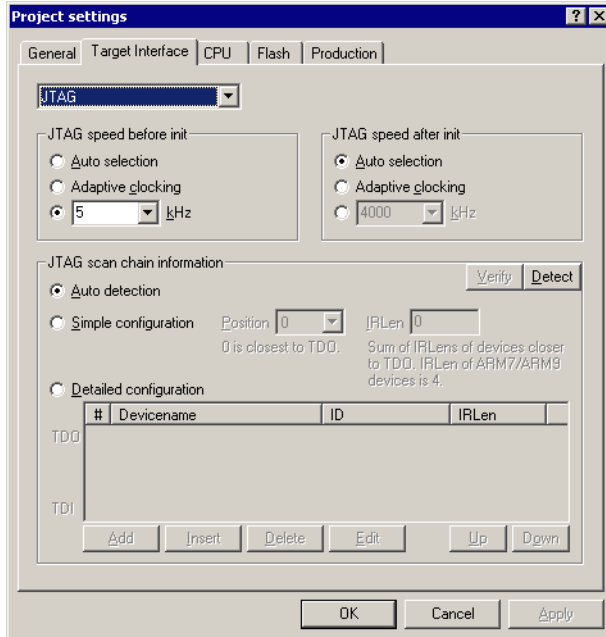

4. Open the **CPU** dialog and select the core architecture in the **Core** choice-list to use J-Flash with an external flash chip. Set the endianess, core ID, RAM address and RAM size of the used MCU.
    To program the internal flash of the chip choose a device from **Device** choice-list. J-Flash uses correct default values (endianess, core ID, RAM address and size) for this device. This is the part where initialization of the external bus interface (if necessary) has to be done. For more information about the valid commands which can be used in an initialization sequence, please refer to *Init steps* on page 39.
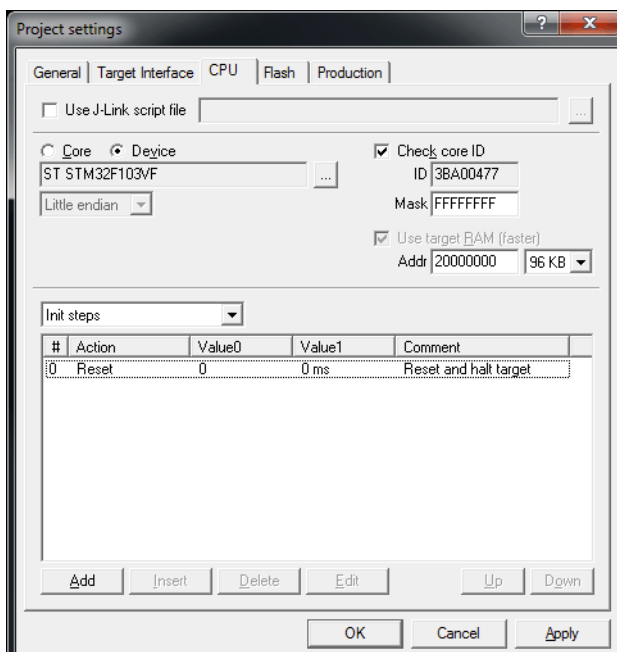
5. The Flash dialog is dependent on selection in the CPU dialog. If you want to pro-
    gram external flash the dialog should look similar to the screenshot below. The
    used flash chip can be automatically detected or chosen from a list if you disable
    the Automatic detection checkbox. If you choose the **Automatic detection** fea-
    ture, the only required settings are the **Base Addr**, **Organization** and **Chip(s)**
    fields.
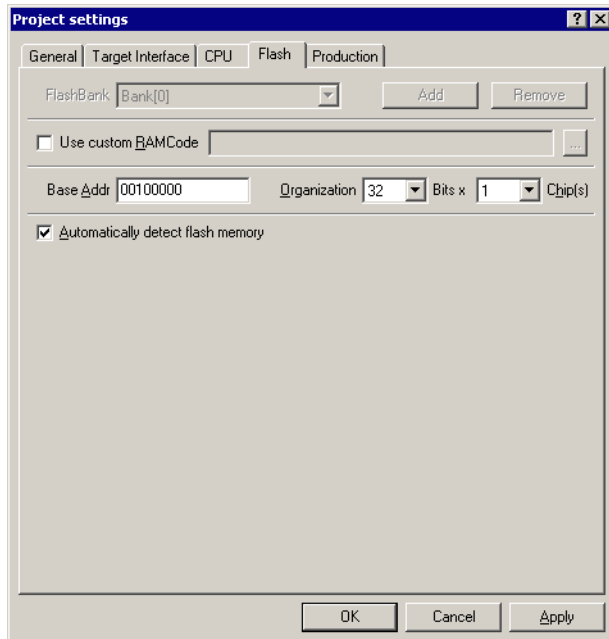
    If you want to program the internal flash of an MCU the dialog should look similar
    to the screenshot below. Normally, all default settings can be used without modi-
    fications.
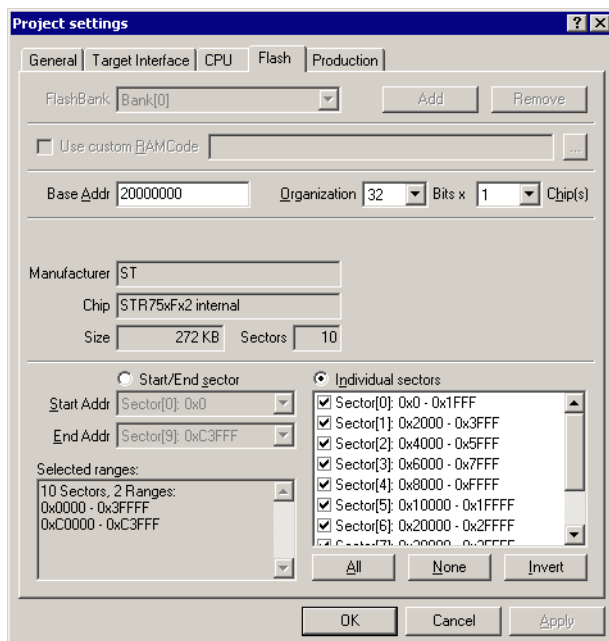
6. In the **Production** dialog is secondary for a setup. You can define the behaviour of the Auto option (**Target** -> **Auto** or shortcut: F7).
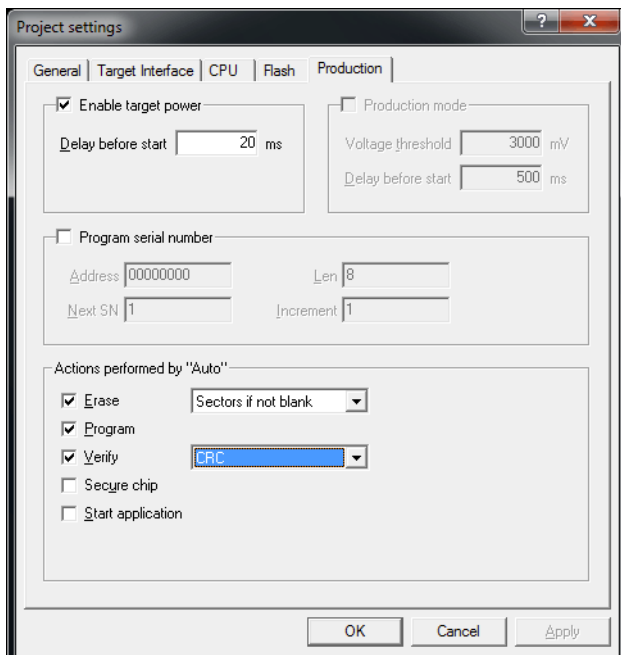

7. Save your project (**File** -> **Save Project**) and test it.

6.2   Creating a new init sequence

Many microcontrollers require a custom init sequence to initialize the target hard-ware, for example the initialize the PLL, disable the watchdog or define the wait states of the flash. This means that you have to build an compatible init sequence for the microcontroller, if you create a new project or modify one of the existing projects.

You can build or update a custom init sequence in the **CPU** dialog in the **Project settings** menu. Click the **Add** button to open the **Add action** dialog.
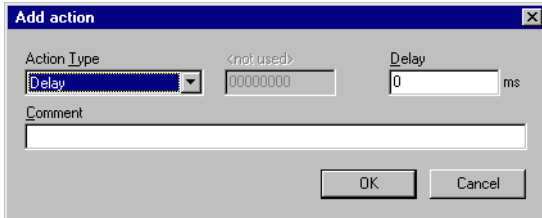
In the **Action Type** choice-list all possible commands are listed. The following two textboxes are dependent on the chosen command. They are grayed out or used to enter the required parameter. The **Comment** textbox should be used to enter a short description of the action. For a list of all valid commands which can be used in an init sequence, please refer to *Init steps* on page 39.

6.2.1   Example init sequence

A good example of a typical init sequence is the init sequence of an AT91SAM7 CPU. The following example is excerpted from the J-Flash project for the AT91SAM7S256.

The example init sequence step by step

0.   Reset the target with J-Link reset strategy 0 and 0 delay.
1.   Disable the watchdog by writing to the Watchdog Timer Mode Register.
2.   Set flash wait states by writing to the MC Flash Mode Register.
3.   Set the PLL by writing to power management controller.
4.   Set a delay of 200ms.
5.   Set the PLL and the divider by writing to PLL Register of the power management controller.
6.   Set a delay of 200ms.
7.   Set the master and processor clock by writing to the Master Clock Register of the power management controller.

The steps implemented in J-Flash:
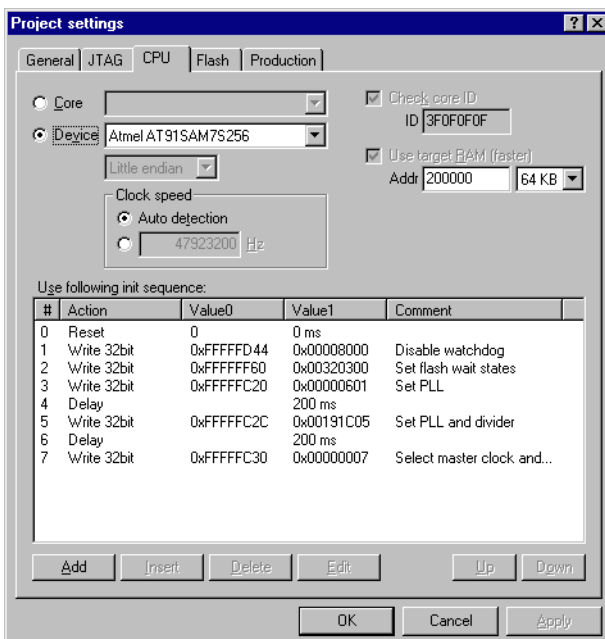
6.3   Serial number programming

J-Flash 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 in the working folder (depending on the configuration) need to be configured first.
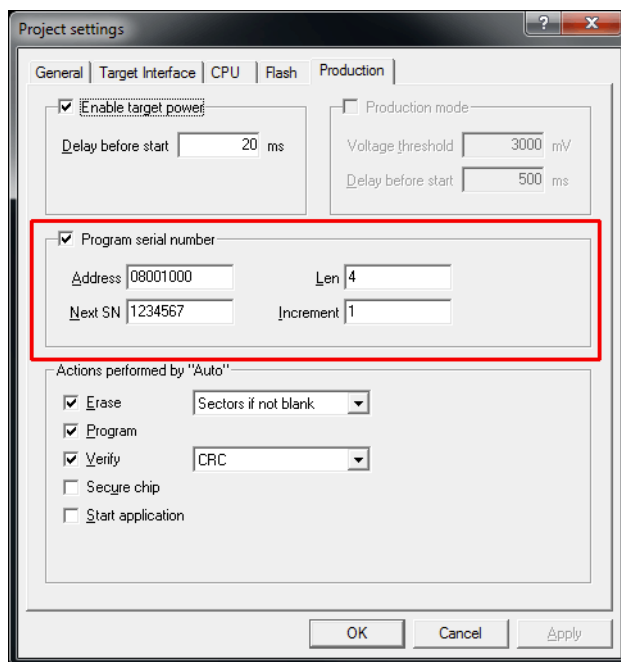
In general, J-Flash supports two ways of programming a serial number into the target:

1. Programming continuous serial numbers. Serial number is 1-4 bytes in size. Start serial number, increment, serial number size and address have to be configured in the J-Flash project production settings.
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 sitze and address is configured in J-Flash production project settings. Serial number list file needs to be specified and created by user.

In the following, some generic information how to setup a serial number programming configuration are given.

6.3.1   Serial number settings

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

Setting	Meaning
Address	The address the serial number should be programmed at.
Len	The length of the serial number (in bytes) which should be programmed. If no serial number list file is given, J-Flash allows to use a 1-4 byte serial number. In case of 8 is selected as length, the serial number and its complementary is programmed at the given address. In case a serial number list file is given, J-Flash will take the serial number bytes from the list file. If a serial number in the list file does not define all bytes of Len, the remaining bytes are filled with 0s. No complements etc. are added to the serial number.
Next SN	In case no serial number list file is given, Next SN is next serial number which should be programmed. The serial number is always stored in little endian format in the flash memory. In case a serial number list file is given, Next SN describes the line of the serial number list file where to read the next serial number bytes from. J-Flashs starts counting with line 0, so in order to start serial number programming with the first line of the SNList.txt, Next SN needs to be set to 0.
Increment	Specifies how much Next SN is incremented.

Table 6.1: J-Flash serial number settings

6.3.2   Serial number file

When starting the program process **Target -> Auto**, J-Flash will create a serial number file named as <JFlashProjectName>_Serial.txt. 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.

6.3.3   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 **Target -> Auto**, J-Flash will check for a serial number list file named as <JFlashProjectName>_SNList.txt in the directory where the J-Flash project is located. 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.
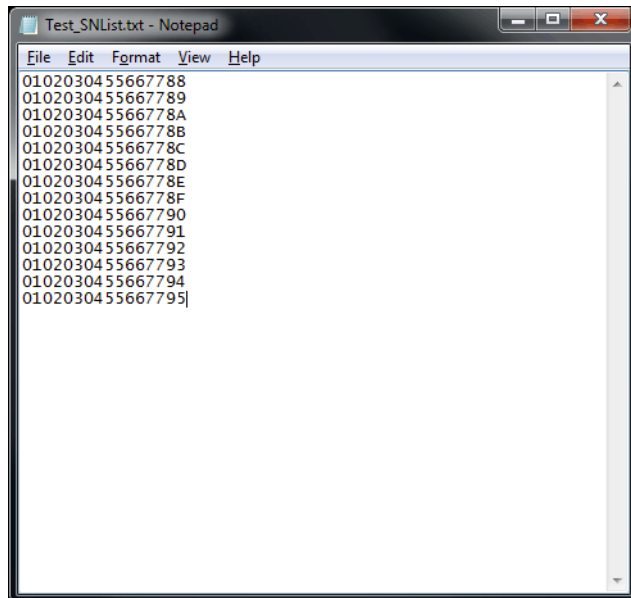
Example

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 59.

Which line J-Flash 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 59. 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 ARM.

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 J-Flash

Note: When using Windows 7, please make sure that the used project file is located at a folder with write permission.

6.3.4   Programming process

J-Flash will increment the serial number in `<JFlashProjectName>_Serial.txt` by the value defined in `Increment`, after each successful programming cycle.

6.3.5   Sample setup

In the following a small sample is given how to setup J-Flash 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, start value and increment

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

- *Address* is 0x08001000
- *Len* is 4 (bytes)
- *Next SN* is 1234567
- *Increment* is 1
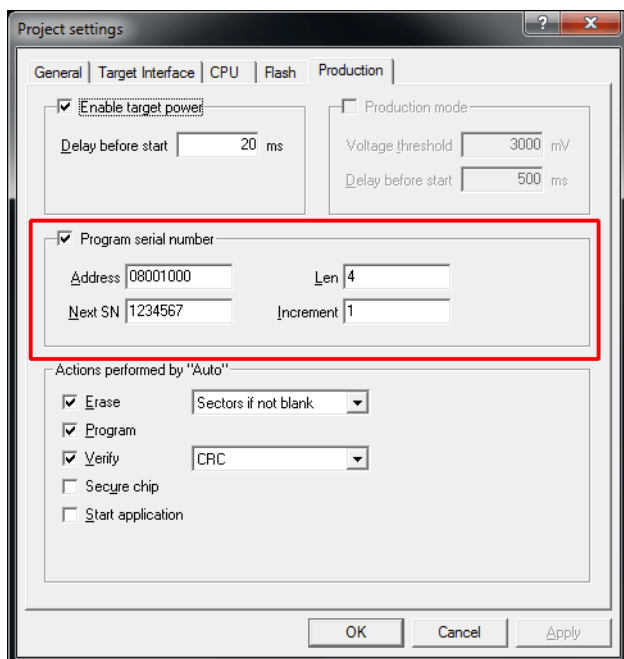
Now J-Flash is prepared to program the 8-byte serial number.

After programming the serial number, J-Flash creates the `<JFlashProject-Name>_Serial.txt`.

Chapter 7

Device specifics

This chapter gives some additional information about specific devices.

7.1   Analog Devices

J-Flash supports flash programming for the Analog Devices ADuC7xxx core family.

7.1.1   ADuC7xxx

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Flash and check the available projects. If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to *J-Link / J-Trace User Guide* for device specifics which are not related to flash programming.

7.2 ATMEL

J-Flash supports flash programming for the ATMEL AT91SAM7 core family.

7.2.1 AT91SAM7

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Flash and check the available projects. If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to *J-Link / J-Trace User Guide* for device specifics which are not related to flash programming.

7.2.2 AT91SAM9

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Flash and check the available projects. If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to *J-Link / J-Trace User Guide* for device specifics which are not related to flash programming.

7.3   Freescale

7.3.1   MC13224

J-Flash comes with a sample project as well as a RAMCodeV2 allowing to program the internal SPI flash of the Freescale MC13224 device. Please note that the device needs some special handling which is in customer's responsibility:

1. The first 4 bytes of the flash are reserved for a validation word that is used during boot process to check if the flash contains a valid application. The validation word can be: "OKOK" (0x4F 0x4B 0x4F 0x4B) or "SECU" (0x53 0x45 0x43 0x55) It is user's responsibility to ensure that the data file he downloads to flash memory contains this validation word.
   Note: If "SECU" is chosen, the JTAG interface is disabled, the device is secured.
2. The next four bytes after the validation word are reserved for a data-word that used to determine how many bytes of flash content shall be copied to RAM when the device boots. A maximum of 96 KBytes can be copied to RAM (This is the whole RAM size).
   Example: Your application in flash has 32 KBytes in size, then the data-word would be 0x8000 (32 KBytes). It is stored in little endian format. When viewing the data file in J-Flash, it would look like as follows:
   00000: 4F 4B 4F 4B 00 80 00 00 XX XX XX XX XX XX XX XX (4-bytes "OKOK", 4-bytes data word, followed by data)

7.3.2   MPC560

J-Flash supports flash programming of the internal code flash, the shadow-area and internal data flash for the Freescale MPC560 device family. The internal code and data flash can be programmed without further attention.

Programming the shadow-area must be done carefully as this flash area includes the censorship values which are used to configure the "Censored mode". Depending on the censorship value, the debug logic can be disabled. The erased state (0xFF) of all the volatile censored mode registers is the protected state. So erasing but not programming the censored mode registers will protect the device.

Therefore, J-Flash comes with two different device options for each device. One which allows programming the internal code and data area, only in order prevent users to lock their devices by chance. One which allows programming those areas and the shadow-area as well. The second configuration, is marked by the extension "(allow shadow)".

Programming the shadow-area

First, the "MPC560xx (allow shadow)" device must be selected in order to be able to touch the shadow-area. Even when selecting the (allow shadow) variant, the shadow-area is not touched / erased if not necessary (data file does not contain data in the shadow-area). When performing chip-erase, J-Flash reads back the censorship registers, erase the shadow-flash and reprogram the censorship registers in order to guarantee that the device will not be locked.

In case, the data image contains data in the shadow-area, J-Flash erases the entire shadow-area block and program it with the data from the user data image. In this case, J-Flash does not restore the censorship registers as it assumes that touching the area is done on purpose.

So when programming the shadow-area, make sure to program the censorship registers with correct values (in case of doubt: reset values) as well in order to ensure that the device will not be locked.

Note: When programming the shadow-area, the image must contain data for the entire area, even if the data to be programmed are 0xFF.

7.4　NXP

J-Flash supports flash programming for the NXP LPC core family.

7.4.1　LPC2xxx

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Flash and check the available projects. If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to *J-Link / J-Trace User Guide* for device specifics which are not related to flash programming.

7.5   OKI

J-Flash supports flash programming for the OKI ML67Q40x core family.

7.5.1   ML67Q40x

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Flash and check the available projects. If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to *J-Link / J-Trace User Guide* for device specifics which are not related to flash programming.

7.6   ST Microelectronics

J-Flash supports flash programming for the ST Microelectronics STR71x, STR73x, STR75x, STR91x, PPC and the Cortex-M3 core families.

7.6.1   SPC560

J-Flash supports flash programming of the internal code flash, the shadow-area and internal data flash for the ST SPC560 device family. The internal code and data flash can be programmed without further attention.

Programming the shadow-area must be done carefully as this flash area includes the censorship values which are used to configure the "Censored mode". Depending on the censorship value, the debug logic can be disabled. The erased state (0xFF) of all the volatile censored mode registers is the protected state. So erasing but not programming the censored mode registers will protect the device.

Therefore, J-Flash comes with two different device options for each device. One which allows programming the internal code and data area, only in order prevent users to lock their devices by chance. One which allows programming those areas and the shadow-area as well. The second configuration, is marked by the extension "(allow shadow)".

Programming the shadow-area

First, the "SPC560xx (allow shadow)" device must be selected in order to be able to touch the shadow-area. Even when selecting the (allow shadow) variant, the shadow-area is not touched / erased if not necessary (data file does not contain data in the shadow-area). When performing chip-erase, J-Flash reads back the censorship registers, erase the shadow-flash and reprogram the censorship registers in order to guarantee that the device will not be locked.

In case, the data image contains data in the shadow-area, J-Flash erases the entire shadow-area block and program it with the data from the user data image. In this case, J-Flash does not restore the censorship registers as it assumes that touching the area is done on purpose.

So when programming the shadow-area, make sure to program the censorship registers with correct values (in case of doubt: reset values) as well in order to ensure that the device will not be locked.When programming the shadow-area, the image must contain data for the entire area, even if the data to be programmed are 0xFF.

7.6.2   STM32F10x

7.6.2.1   Securing/Unsecuring the chip

The "Secure Chip" option is available for the STM32 devices. It will read-protect the internal flash memory of the STM32. J-Flash is also able to unsecure a read-protected STM32 device.

Note: Unsecuring a read protected device will cause a mass erase of the flash memory.

7.6.2.2   Option byte programming

J-Flash supports programming of the option bytes for STM32 devices. In order to program the option bytes simply choose the appropriate Device, which allows option byte programming, in the CPU settings tab (e.g. `STM32F103ZE (allow opt. bytes)`). J-Flash will allow programming a virtual 16-byte sector at address

0x06000000 which represents the 8 option bytes and their complements. You do not have to care about the option bytes' complements since they are computated automatically. The following table describes the structure of the option bytes sector

Address	[31:24]	[23:16]	[15:8]	[7:0]
0x06000000	complement	Option byte 1	complement	Option byte 0
0x06000004	complement	Option byte 3	complement	Option byte 2
0x06000008	complement	Option byte 5	complement	Option byte 4
0x0600000C	complement	Option byte 7	complement	Option byte 6

Table 7.1: Option bytes sector description

Note: Writing a value of 0xFF inside option byte 0 will read-protect the STM32. In order to keep the device unprotected you have to write the key value 0xA5 into option byte 0.

Note: The address 0x06000000 is a virtual address only. The option bytes are originally located at address 0x1FFFF800. The remap from 0x06000000 to 0x1FFFF800 is done automatically by J-Flash.

Example

To program the option bytes 2 and 3 with the values 0xAA and 0xBB but leave the device unprotected your option byte sector (at addr 0x06000000) should look like as follows:

Address	[31:24]	[23:16]	[15:8]	[7:0]
0x06000000	0x00	0xFF	0x5A	0xA5
0x06000004	0x44	0xBB	0x55	0xAA
0x06000008	0x00	0xFF	0x00	0xFF
0x0600000C	0x00	0xFF	0x00	0xFF

Table 7.2: Option bytes programming example

For a detailed description of each option byte, please refer to *ST programming manual PM0042, section "Option byte description".*

7.6.3   STM32F2 series devices - option byte programming

Please refer to *STM32F4 series devices - option byte programming* on page 70.

7.6.4   STM32F4 series devices - option byte programming

The STM32F2 series devices provide some option bytes which allow some "permanent" configuration as well as readout protection for the device. Unfortunately, the option bytes become effective immediately when programmed. This means when enabling the read protection of the device via option bytes, the programmer will immediately lose the access to the flash, without any possibility to verify the complete flash operation. Therefore, the option bytes cannot be programmed as part of the flash image.

In order to program the option bytes, some steps need to be added in J-Flash at **Options** -> **Project settings** -> **CPU** -> **Exit Steps**.
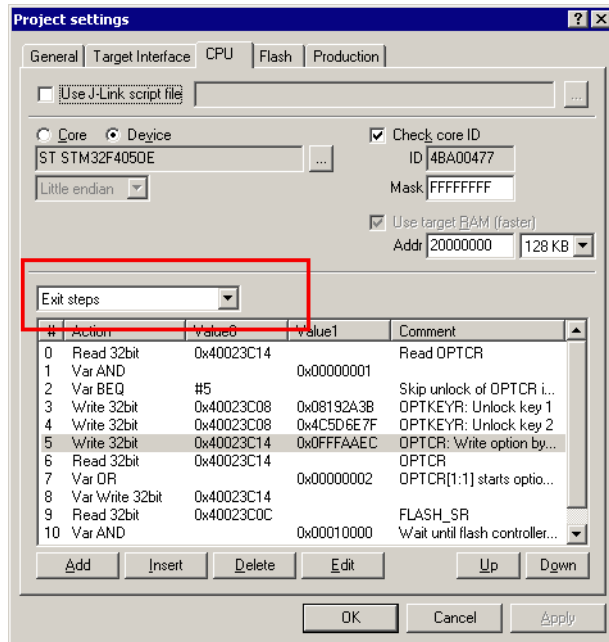

These steps will be executed after **Target** -> **Auto** has completed programming the image.

The option byte values are transmitted in the step which writes the OPTCR.

J-Flash comes with an example project for the STM32F405OE which demonstrates how to program the option bytes. These steps can be adapted for all STM32F2 / F4 devices since they are identical.

The example can be found by selecting Create project from template, navigating to the ST folder and select STM32F405OE_OptBytes.jflash.
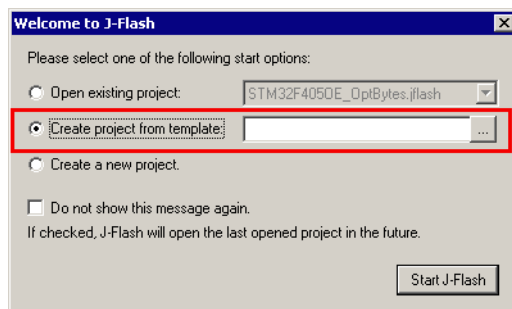
7.6.5   STR 71x

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Flash and check the available projects. If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to *J-Link / J-Trace User Guide* for device specifics which are not related to flash programming.

7.6.6   STR 73x

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Flash and check the available projects. If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to *J-Link / J-Trace User Guide* for device specifics which are not related to flash programming.

7.6.7   STR 75x

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Flash and check the available projects. If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to *J-Link / J-Trace User Guide* for device specifics which are not related to flash programming.

7.6.8   STR91x

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Flash and check the available projects. If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to *J-Link / J-Trace User Guide* for device specifics which are not related to flash programming.

7.7 Texas Instruments

J-Flash supports flash programming for the TI TMS470 core family.

7.7.1 TMS470

J-Flash includes "ready-to-use" projects for all supported devices. For a complete list of supported devices, open J-Link RDI configuration dialog and check the device list of the Flash programming tab (refer to Flash configuration on page 38 for detailed information). If you miss the support of a particular device, do not hesitate to contact Segger.

Refer to J-Link / J-Trace User Guide for device specifics which are not related to flash programming.

Chapter 8

Target systems

The following chapter lists all supported flash devices.

8.1   Which devices can be programmed by J-Flash?

J-Flash supports programming of internal and external flash devices. The external flash device can be:

- Parallel NOR flash
- Serial NOR flash
- NAND flash
- DataFlash

For parallel NOR flash any combination of ARM CPU and parallel NOR flash device (1x8bit, 2x8bit, 4x8bit, 1x16bit, 2x16bit, 1x32bit) is supported, if the NOR flash device is CFI-compliant. If the NOR flash device which is used is not CFI-compliant, you have to explicitly select the flash device in J-Flash. For a list of all parallel NOR flash devices which can be explicitly selected in J-Flash, please refer to *Supported Flash Devices* on page 78.

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. J-Flash 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.*

For more information about which which microcontrollers with internal flash are supported by J-Flash, please refer to *Supported microcontrollers* on page 77.

SEGGER is constantly adding support for new devices. If you need support for a chip or flash not listed in the tables, do not hesitate to contact us.

8.2   Supported microcontrollers

J-Flash supports download into the internal flash of a large number of microcontrollers. You can always find the latest list of supported devices on our website:

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

8.3   Supported Flash Devices

J-Flash supports a large number of external parallel NOR flash devices. In general, every CFI-compliant parallel NOR flash device is supported by J-Flash. For non-CFI compliant ones, J-Flash allows the user to explicitly select the device. You can always find the latest list of supported devices on our website:

http://www.segger.com/supported-flash-devices.html

Chapter 9

Performance

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

9.1   Performance of MCUs with internal flash memory

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

Microcontroller	Size [kByte]	Program time [sec]	Program speed [kB/sec]	Erase Time [sec]	Erase speed [kB/sec]
Analog Devices ADuC7020	62	2.234	27.752	3.031	20.455
Atmel AT91SAM7S64	64	3.235	19.783	- Not required	
Atmel AT91SAM7S256	256	6.734	38.016	- Not required	
NXP LPC2148	500	3.953	126.486	12.312	40.610
NXP LPC2138	500	3.906	128.008	12.312	40.610
NXP LPC2129 V1	248	1.828	135.667	7.812	31.746
NXP LPC2106	120	0.948	126.582	6.875	17.454
NXP LPC2129 V2	248	1.797	138.007	7.750	32.000
NXP LPC2294	248	1.875	132.266	7.812	31.746
ST STR711	272	4.890	55.623	9.703	28.032
ST STR912	512	7.000	73.142	9.375	54.613
TI TMS470R1B1M	1024	10.953	93.490	18.359	55.776

Table 9.1: List of performance values of MCUs with internal flash

9.2   Performance of MCUs with external flash memory

Hardware	Flash device	Organization	Speed
Atmel AT91EB40	Atmel AT49BV162A	1*16 Bits	105.025 kB/s
Cogent CSB337	Intel 28F640J3	1*16 Bits	93.058 kB/s
NetSilicon NS9360	AMD Am29LV160DB	2*16 Bits	185.171 kB/s
Logic LH7A400	Intel 28F640J3A120	2*16 Bits	154.978 kB/s

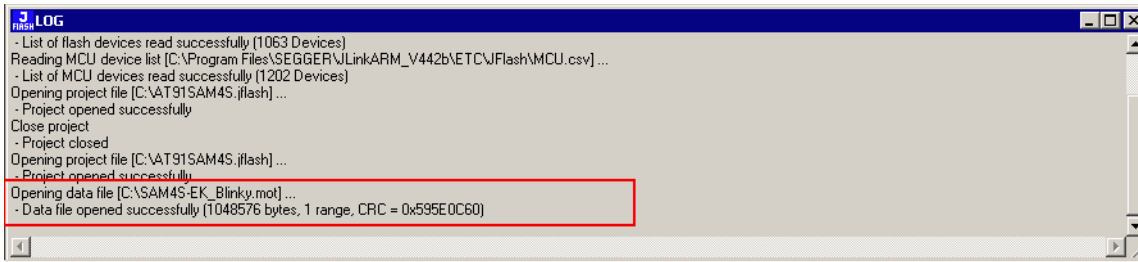**Table 9.2: List of performance values of MCUs with external flash**

Chapter 10

Background information

This chapter provides some background information about specific parts of the J-Flash software.

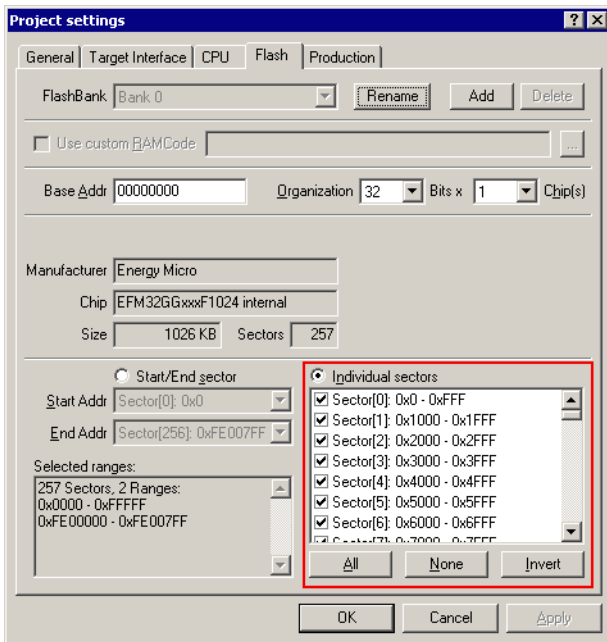# 10.1   CRC of current data file

When opening a data file in J-Flash (**File->Open...**), J-Flash calculates and displays the CRC of the user data in this file.

The following steps are taken into consideration when calculating this CRC:

1. The CRC is calculated over all sectors which are selected in the current project

2. Everything that is not covered by the data file (gaps in the data file, unused sectors etc.) which is opened, is assumed as 0xFF during the CRC calculation.
3. The polynomial which is used for the CRC calculation is 0xEDB88320.

Chapter 11

Support

The following chapter provides information about how to contact our support.

11.1 Troubleshooting

11.1.1 General procedure

- Make sure your J-Link is working as expected. See the troubleshooting section in the J-Link manual.
- Ensure that the target hardware matches the project file settings. Pay special attention to the following aspects:
  - Init sequence
  - Clock speed
  - RAM address
  - Flash base address
  - MCU / Flash chip
  - Flash organization
- Try to program your target device using a sample project file if available. J-Flash ships with an extensive number of project files for many target boards. See section *Sample Projects* on page 24 for a complete list of project files.
- The JTAG clock frequency depends on several factors, e.g. cable length, target board etc. Try setting the frequency to lower or higher values accordingly.
- Make sure the flash memory is unlocked before programming or erasing.

11.1.2 Typical problems

Failed to connect

Meaning:

This error message is shown if any error occurs during the connection process.

Remedy:

First of all, make sure the target is actually connected to J-Link. Verify the correctness of the init sequence, check the JTAG speed, and ensure the correct flash type is selected.

Programming / Erasing failed

Meaning:

The flash memory sector may be locked and programming or erasing the respective memory section fails therefore.

Remedy:

Make sure the memory sector is unlocked before programming or erasing. J-Flash provides a dedicated menu item for unlocking flash memory.

Timeout errors during programming

Meaning:

A timeout occurs if the target is too slow during DCC communication or the target flash memory is too slow during programming.

Remedy:

Using smaller RAM block sizes may fix this problem.

Blank check failed

Meaning:

The target memory was not empty during blank check.

Remedy:

Erase target memory.

RAM check failed

Meaning:

No RAM found at the specified RAM location.

Remedy:

Make sure a correct RAM address is specified in the project settings. See section *CPU Settings* on page 37.

Unexpected core ID

Meaning:

The specified CPU core ID does not match with the one read from the target CPU.

Remedy:

Ensure the specified core ID is correct for the used target CPU. See section *CPU Settings* on page 37 for information about setting the core ID.

Unsupported flash type / bus width

Meaning:

The target flash memory or the bus organization is not yet supported.

Remedy:

Inform us about the flash type you want to use. SEGGER is constantly adding support for new flash memory devices.

No matching RAMCode

Meaning:

There is no programming algorithm available for the selected target memory type.

Remedy:

Inform us about the flash type you want to use. SEGGER is constantly adding support for new flash memory devices.

11.2 Contacting support

If you experience a J-Flash related problem and the advices from the sections above do not help you to solve it, you may contact our J-Flash support. In this case, please provide us with the following information:

- A detailed description of the problem.
- The relevant log file and project file. In order to generate an expressive log file, set the log level to "All messages" (see section *Global Settings* on page 45 for information about changing the log level in J-Flash).
- The relevant data file as a .hex or .mot file (if possible)
- The processor and flash types used

Once we received this information we will try our best to solve the problem for you. Our contact address is as follows:

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

Index