J-Flash

User guide of the stand-alone flash programming software

Document: UM08003
Software Version: 8.74
Date: September 30, 2025

A product of SEGGER Microcontroller GmbH

www.segger.com

Disclaimer

The information written in this document is assumed to be accurate without guarantee. The information in this manual is subject to change for functional or performance improvements without notice. SEGGER Microcontroller GmbH (SEGGER) assumes no responsibility for any errors or omissions in this document. SEGGER disclaims any warranties or conditions, express, implied or statutory for the fitness of the product for a particular purpose. It is your sole responsibility to evaluate the fitness of the product for any specific use.

Copyright notice

Trademarks

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

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

Contact address

SEGGER Microcontroller GmbH

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

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

Manual versions

This manual describes the current software version. If you find an error in the manual or a problem in the software, please report it to us and we will try to assist you as soon as possible.

Contact us for further information on topics or functions that are not yet documented.

Print date: September 30, 2025

Manual version	Revision	Date	By	Description
7.86	0	230223	JB	Chapter “Command Line interface” * Section “Command line options”: Parameters do not necessarily have to follow a command directly anymore, a space is now also possible.
7.80	0	221005	JB	Chapter “Menu structure” * Section “Help menu elements”: Added additional wiki links and user guides.
7.68	0	220802	LG	Chapter “Command Line interface” * Section “Programming multiple targets in parallel”: Updated batch file example. * Section “Programming multiple targets in parallel”: Added Python script example.
6.70	0	200416	LG	All chapters Updated for new version of J-Flash.
6.70	0	200414	LG	Chapter “Target systems” * Section “Which devices can be programmed by J-Flash?”: Added reference to SEGGER Wiki article about ‘Open Flash Loader’.
6.64	0	200324	LG	All chapters Updated for recently added cross-platform availability of J-Flash.
6.55b	0	191114	FF	Chapter “Settings” * Section “Init steps”: Added info about Verify/Write&Verify.
6.49	4	190814	LG	Chapter “Command Line Interface” * Section “Programming multiple targets in parallel”: Updated batch scripts.
6.48b	0	190812	AB	Chapter “Command Line Interface” * Section “Command line options”: Updated CL option -jlinkdevicesxml-path.
6.32	4	190418	FF	Chapter “Getting Started” * Section “Menu structure”: added a note in the table “Target menu elements” for Manual Programming > Program.
6.32	3	190327	MF	Chapter “Settings” * Section “Production setting”: added clarification note for target power supply.
6.32	2	180621	LG	Chapter “Command Line Interface” * Section “Command line options”: Added new CL option -setcpuidcode.
6.32	1	180427	LG	Chapter “Command Line Interface” * Section “Command line options”: Added new CL option -saveas. * Section “Command line options”: Updated CL option -merge. * Section “Command line options”: Updated CL option -setrxidcode.
6.30	2	180417	NV	Screenshots updated. Chapter “Command Line Interface” * Section “Command line options”: Added new CL option -hide. * Section “Command line options”: Updated CL option -jflashlog. * Section “Command line options”: Updated CL option -jlinklog.
6.30	1	180216	LG	Screenshots updated. Chapter “Settings” * Added section “Performance settings”
5.02c	0	150914	RH	Chapter “Command Line Interface” * Section “Command line options” Added new commands: -verifycrca, verifycrcs and verifycrcc
5.02a	0	150907	EL	Chapter “Command Line Interface” * Section “Command line options” updated.
5.00c	0	150611	EL	Chapter “Command Line Interface” * Section “Programming multiple targets in parallel” updated.

Manual version	Revision	Date	By	Description
4.98	2	150427	EL	Chapter "Command Line Interface" * Section "Command line options" Added new command: -ip and -USB.
4.98	1	150320	AG	Chapter "Background information" * Section "CRC of current data file" Polynomial corrected.
4.98	0	150113	NG	Chapter "Command Line Interface" Changed "JFlashARM.exe" to "JFlash.exe".
4.96	0	150109	EL	Chapter "Device specifics" * Section "ST" updated. * Section "Freescale" updated.
4.82	0	140307	AG	Chapter "Device specifics" * Section "ST" updated.
4.80	0	131220	AG	Chapter "Command Line Interface" * Section "Command line options" updated.
4.73c	0	130703	JL	Chapter "Getting Started" * Added Section "Start Dialog"
4.66	1	130320	EL	Chapter "Settings" * Section "CPU Settings" Added description for the core ID "Mask" field
4.66	0	130221	JL	Chapter "Introduction" * Section "What is J-Flash" Added Linux and Mac OSX
4.58	0	121113	JL	Chapter "Command Line Interface" * Section "Batch processing" updated. * Section "Command line options" updated.
4.52	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	0	120724	EL	Chapter "Create a new J-Flash project" * Section "Configuration for serial number programming" added.
4.42b	0	120217	AG	Chapter "Background information" * Section "CRC of current data file" added.
4.24	0	110216	AG	Chapter "Target systems" updated.
4.16	1	100817	AG	Chapter "Command Line Interface" * Section "Command line options" corrected.
4.16	0	100723	KN	Chapter "Settings" * Section "Init sequence" updated.
4.10	4	091204	AG	Chapter "Device specifics" * Section "Freescale" added.
4.10	2	090918	AG	Chapter "Command Line interface" * Section "Command line options" updated.
4.10	1	090902	AG	Chapter "Device specifics" * Section "ST Microelectronics" updated.
4.10	0	090825	AG	Chapter "Device specifics" * Section "ST Microelectronics" updated.
4.04	1	090414	AG	Chapter "Introduction" * Section "What is J-Flash?" updated.
4.04	0	090204	AG	Chapter "Command Line Interface" * Section "Overview" updated. * Section "Command Line Options" updated.
3.97e	0	081204	KN	Chapter "Target systems" * Section "Supported Flash Devices" updated Chapter "Settings" * Section "Init sequence" corrected

Manual version	Revision	Date	By	Description
3.91n	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	0	080811	AG	Chapter “Targets” * Section “Supported Microcontrollers” updated.
3.80	2	080408	AG	Chapter “Licensing” * Section “Introduction” added. * Section “License types” added.
3.80	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	0	080206	SK	Chapter “Device specifics” added. Chapter “Target systems” * Section supported MCUs updated.
3.68	1	070508	SK	Chapter “Installation” updated. Chapter “Command Line Interface”: * Section “Batch processing” added. Various improvements.
3.66	1	070322	SK	Chapter “Target systems” updated. Chapter “Getting started” updated.
3.46	4	061222	SK	Section “About” and company description added.
3.46	3	061124	OO	Chapter “Performance” updated.
3.46	2	061121	OO	Chapter “Performance” updated.
3.46	1	060929	TQ	Update supported target devices.
3.42	1	060912	TQ	Update supported target devices.
3.36	1	060801	TQ	Update supported target devices.
3.24	1	060530	TQ	Update supported target devices.
3.00	2	060116	OO	Screenshots updated.
3.00	1	060112	TQ	Nothing changed. Just a new software version.
2.14	0	051025	TQ	Update supported target devices.
2.10	0	050926	TW	Added troubleshooting section.
2.04	0	050819	TQ	Nothing changed. Just a new software version.
2.02	0	050808	TW	Command line added.
2.00	0	050707	TW	Initial Version

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 Ritchie (ISBN 0--13--1103628), which describes the standard in C programming and, in newer editions, also covers the ANSI C standard.

How to use this manual

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

Typographic conventions for syntax

This manual uses the following typographic conventions:

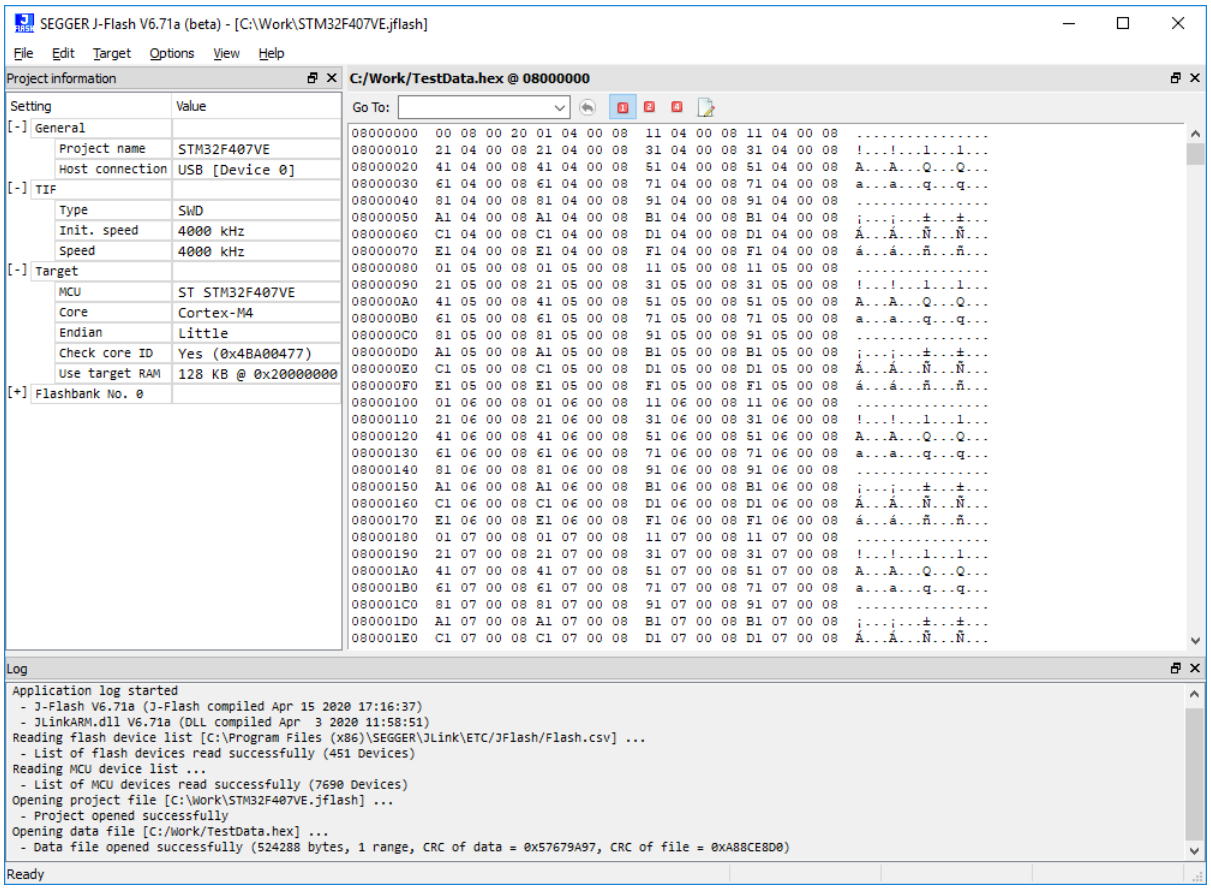
Style	Used for
Body	Body text.
`Keyword`	Text that you enter at the command prompt or that appears on the display (that is system functions, file- or pathnames).
`Parameter`	Parameters in API functions.
`Sample`	Sample code in program examples.
`Sample comment`	Comments in program examples.
Reference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

Table of contents

Chapter 1

Introduction

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



J-Flash main window

1.1 What is J-Flash?

J-Flash is a stand-alone flash programming software for PCs running Windows, Linux or macOS.

J-Flash has an intuitive user interface and makes programming flash devices convenient. J-Flash requires a J-Link / Flasher, to interface to the hardware. It is able to program internal and external flash at very high speeds, upwards of 550 KBytes/s depending on the chip. 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/ARM9/ARM11, Cortex-M0/M1/M3/M4/M7, Cortex-A5/A8/A9/R4/R5 and Renesas RX600 chip makes it a great solution for most projects.

1.1.1 Supported Operating Systems

The following operating systems are supported:

- Microsoft Windows (x86 / x64 / Arm64)
- Linux (x86 / x64 / Arm64)
- macOS (x64 / Apple M1)

1.1.2 Features

- Any ARM7/ARM9/ARM11, Cortex-M0/M1/M3/M4/M7, Cortex-A5/A8/A9/R4/R5 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 54).
- High speed programming: up to 550 KBytes/s* (depending on flash device).
- 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, .bin and .elf support.
- Intuitive user interface.

* = Measured with J-Link V10

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 (UM08001), which describes the device and its use in detail.

1.3 Requirements

- J-Link / Flasher
- Supported operating system (see *Supported Operating Systems* on page 12)
- Interface from Host to probe (USB, Ethernet, WiFi, …)
- Supported device/core (see *Supported microcontrollers* on page 56)

Chapter 2

Licensing

J-Flash may be installed on as many host machines as you want. Without a license key J-Flash can still be used 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, a valid license is required. For an overview which SEGGER products come with a built-in license for J-Flash, please refer to the *J-Link Model overview* . All Flasher models come with a built-in license for J-Flash.

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 J-Flash is described in chapter 2 of the J-Link / J-Trace User Guide (UM08001). The J-Link / J-Trace User Guide (UM08001) is part of the J-Link Software Pack which is available for download under *segger.com/jlink-software.html* .

3.1.1 What is included?

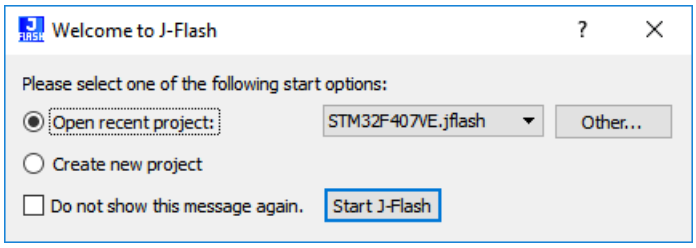
The following table shows the contents of all subdirectories of the J-Link Software Pack with regard to J-Flash:

Directory	Contents
.	The J-Flash application. Please refer to the J-Link Manual (UM08001) 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.
.\Samples\JFlash\ProjectFiles	Contains sample projects with good default settings (see section *Sample Projects* on page 18 for further details).

3.2 Using J-Flash for the first time

3.2.1 Welcome dialog

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



Welcome Dialog

The startup dialog provides the following options:

- ***Open existing project:*** Select a project from the list of recent projects or press **Other...** to open another existing project.
- ***Create new project:*** Opens another dialog to create a new J-Flash project

If *"Do not show this message again."* is checked, J-Flash will execute the option currently selected automatically on future starts without showing the welcome dialog again.

3.2.2 Sample Projects

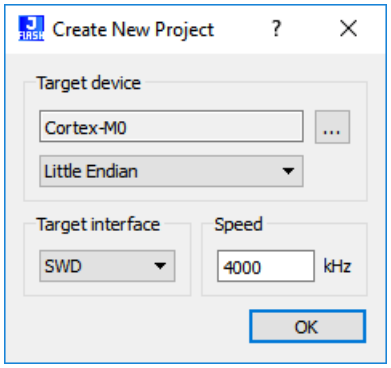
For some setups, special settings / configurations needs to be done in the J-Flash project (e.g. PLL initialization, external bus interface initialization, script files, etc…). Therefore, the J-Link Software Pack already includes some example projects for various special setups which can be used as reference for custom setups.

Those project files can be found in the \Samples\JFlash\ProjectFiles subdirectory of the J-Link Software Pack installation directory.

3.2.3 Creating a new J-Flash project

The recommend way of getting started with J-Flash is to use the **Create New Project** wizard.

- Start by selecting the **Create new project** option inside the Welcome dialog or by selecting **File** -> **New project**
- The new project wizard will launch, which looks like as follows:



New project wizard

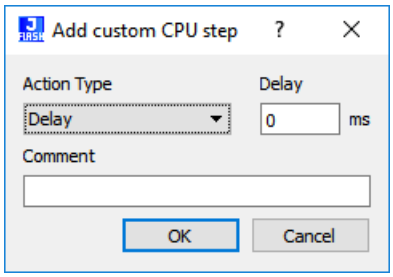
- Select the target device, the target interface and interface speed according to the setup. If only a core is selected, the target endianness must be specified as well.
- Click **OK**

The created Project file is now ready for use. More sophisticated settings can be configured in the **Project settings**. Please refer to *Project Settings* on page 29.

3.2.4 Creating a new init sequence

Many microcontrollers require a custom init sequence to initialize the target hardware, for example to initialize the PLL, disable the watchdog or define the wait states of the flash. This means that an compatible init sequence for the microcontroller must be built, if a new project is created or an existing project is modified.

A custom init sequence can be created or updated in the **Init. steps** tab of the **Project settings** menu. Click the **+** button to open the **Add custom CPU step** dialog.



Init Steps: Add action dialog

In the **Action Type** dropdown menu all available actions are listed. Depending on the type of action, there are either one or two textboxes next to the dropdown menu, which can be used to enter the required parameter. The **Comment** text box should be used to enter a short description of the action. For a list of all valid actions which can be used in an init sequence, please refer to *Init steps* on page 33.

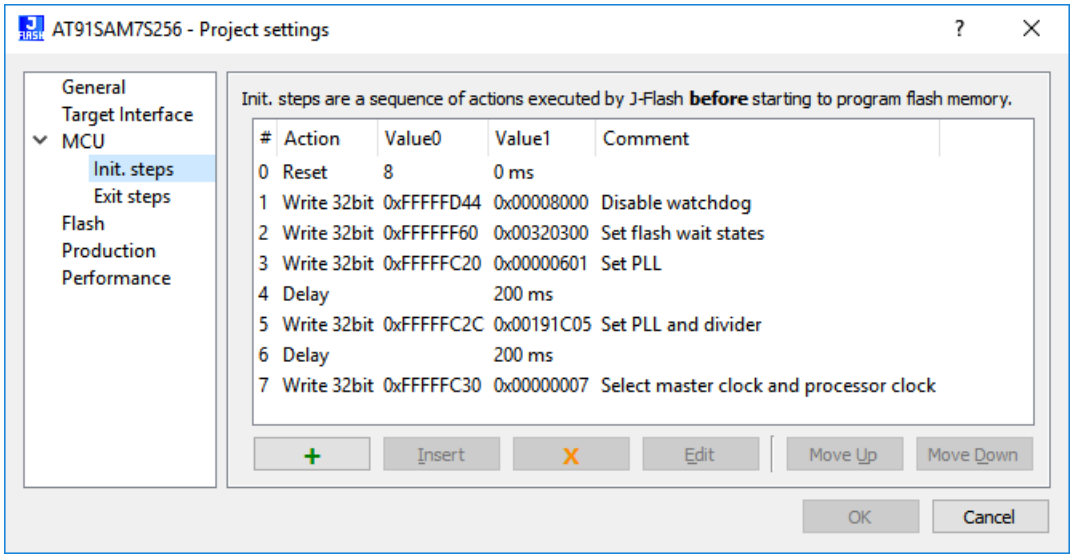
3.2.4.1 Example init sequence

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 8 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:



MCU settings: Init Steps: Example

3.2.5 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.

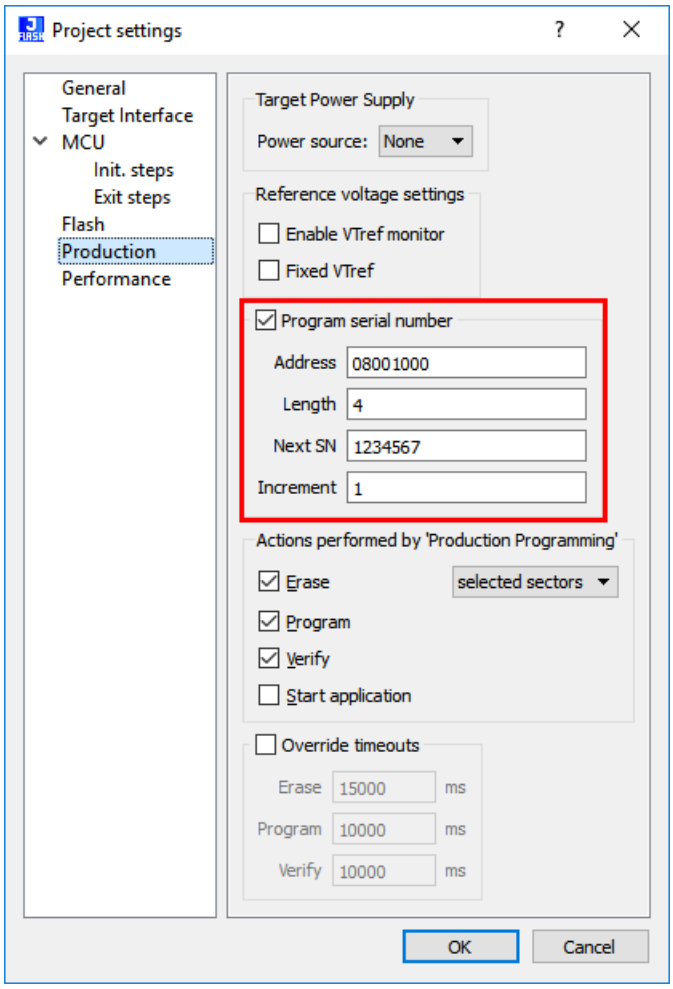
In general, J-Flash supports two ways of programming a serial number into the target:

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 size 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.

3.2.5.1 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:



Program serial number option

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. If 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 the next serial number which should be programmed. The serial number is always stored in little endian format in the flash memory. • In case a serial number list file is given, Next SN describes the line of the serial number list file where to read the next serial number bytes from. J-Flash 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.

3.2.5.2 Serial number file

When starting the program process **Target -> Production Programming**, 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.

3.2.5.3 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 -> Production Programming**, 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.

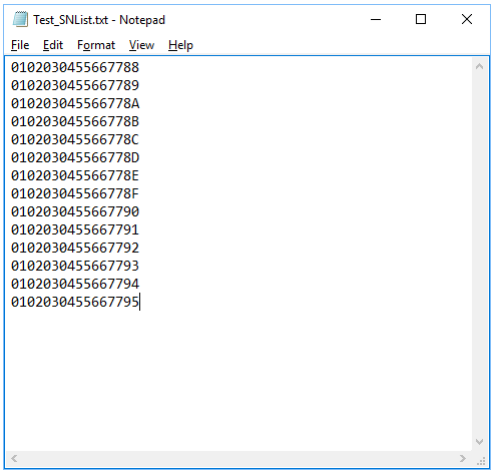
Example

An 8-byte serial number should be programmed at address `0x08000000`. It should be programmed as follows in the memory:

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

The serial number list file should look as follows:

```
0102030455667788
```

SN list example

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 21.

Which line J-Flash will read at the next programming cycle is configured via the Next SN option. For more information, please refer to *Serial number settings* on page 21. In this case Next SN needs to be set to 0, since programming should be started with the serial number bytes defined in the first line of the file.

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

3.2.5.4 Programming process

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

3.2.5.5 Sample setup

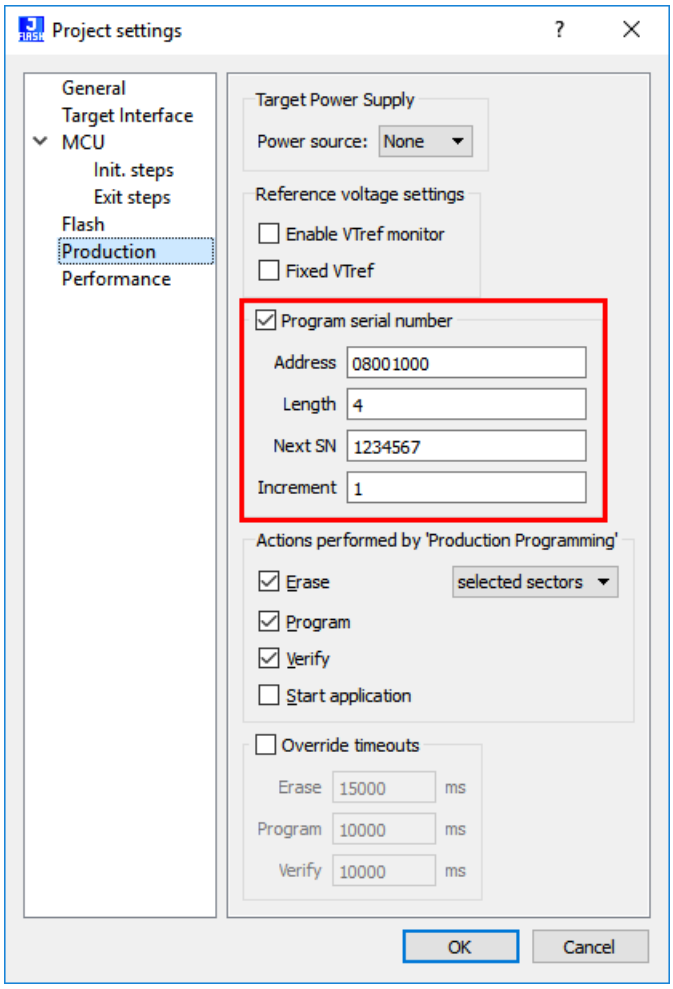
In the following a small sample is given how to setup J-Flash for serial number programming. In the following sample, 4-byte serial numbers starting at 1234567 (`0x12D687`) shall be programmed at address `0x08001000`.

Defining serial number address, length, start value and increment

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

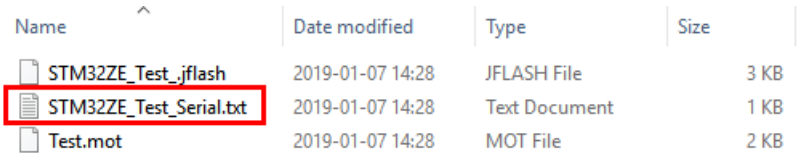
- Address is `0x08001000`
- Len is 4 (bytes)
- Next SN is 1234567

- Increment is 1



Program serial number option

Now J-Flash is prepared to program the 8-byte serial number. After programming the serial number, J-Flash creates the `<JFlashProjectName>_Serial.txt`.



Serial number file

3.3 Menu structure

The main window of J-Flash contains seven dropdown menus (***File***, ***Edit***, ***Target***, ***Options***, ***View***, ***Help***).

File menu elements

Command	Description
Open data file…	Opens a data file that may be used to flash the target device. The data file must be one of the following: Intel HEX file, Motorola S file, Binary file or ELF file (.hex, .mot, .srec, .bin or .elf).
Merge data file…	Merges two data files (.hex, .mot, .srec, .bin or .elf). 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 data file	Saves the data file that currently has focus.
Save data file as…	Saves the data file that currently has focus using the name and location given.
New project	Creates a new project (See *Creating a new J-Flash project*)
Open project	Opens a J-Flash project file. Please 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.
Save Flasher config file…	Saves a .CFG file for stand-alone mode using the name and location given. Please refer to the Flasher documentation (UM08022) for more information regarding stand-alone mode.
Save Flasher data file…	Saves a .DAT file for stand-alone mode using the name and location given. Please refer to the Flasher documentation (UM08022) for more information regarding stand-alone mode.
Download config & data file to Flasher	Prepares a connected Flasher for stand-alone mode using the current project and the data file which had focus most recently. Please refer to the Flasher documentation (UM08022) for more information regarding stand-alone mode.
Download serial number file to Flasher	Downloads a serial number file to a connected Flasher. Please refer to the Flasher documentation (UM08022) for more information regarding stand-alone mode.
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.

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.

Target menu elements

Command	Description
Connect	Creates a connection through the Flasher using the configuration options set in the Project settings… of the Options dropdown menu.
Disconnect	Disconnects a current connection that has been made through the Flasher.
Test > Generate 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.
Test > Test speed	Writes data of an specified size to an defined address, reads the written data back and measures the up- and download speed.
Test > Show CFI info	Reads the CFI query information of a CFI compliant flash device.
Test > Hardware > Activate BUSY	Sets the RS232 Busy signal of a connected Flasher. Can be used to test the RS232 setup.
Test > Hardware > Deactivate BUSY	Resets the RS232 Busy signal of a connected Flasher. Can be used to test the RS232 setup.
Test > Hardware > Activate OK	Sets the RS232 OK signal of a connected Flasher. Can be used to test the RS232 setup.
Test > Hardware > Deactivate OK	Resets the RS232 OK signal of a connected Flasher. Can be used to test the RS232 setup.
Production Programming	Performs a sequence of steps, which can be configured in the Production tab of the Project settings. Additionally, the first step executed are the init steps and the last step executed are the exit steps, which both can be configured in the MCU tab of the project settings.
Manual Programming > Secure Chip	Secures the MCU.
Manual Programming > Unsecure Chip	Unsecures the MCU.
Manual Programming > Check Blank	Checks flash to see if it is empty.
Manual Programming > Erase Sectors	Erases all selected flash sectors.
Manual Programming > Erase Chip	Erases the entire chip.

Command	Description
Manual Programming > Program	Programs the chip using the currently active data file. Please note that no erase / blank check is performed prior programming so the flash is assumed to be in an erased state.
Manual Programming > Program & Verify	Programs the chip using the currently active data file and then verifies that it was written successfully.
Manual Programming > Verify	Verifies the data found on the chip with the data file.
Manual Programming > Read back > Selected Sectors	Reads back the data found in the selected sectors and creates a new data file to store this information.
Manual Programming > Read back > Entire chip	Reads back the data found on the chip and creates a new data file to store this information.
Manual Programming > Read back > Range	Reads back the data found in a range specified by the user and creates a new data file to store this information.
Manual Programming > Start Application	Starts the application.

Options menu elements

Command	Description
Project settings…	Opens the project settings dialog.
Global settings…	Opens the global settings dialog.

View menu elements

Command	Description
Show project information	Opens and/or sets the focus to the project window.
Show log	Opens and/or sets the focus to the log window.

Help menu elements

Command	Description
J-Flash Wiki	Opens the J-Flash wiki in a browser.
J-Flash SPI Wiki	Opens the J-Flash SPI wiki in a browser.
J-Link Wiki	Opens the J-Link wiki in a browser.
Flasher Wiki	Opens the Flasher wiki in a browser.
J-Flash User Guide	Opens this document in a .PDF application.
J-Link User Guide	Opens the J-Link Manual (UM08001) in a .PDF application.
Flasher User Guide	Opens the Flasher Manual (UM08022) in a .PDF application.
Flasher ATE User Guide	Opens the Flasher ATE Manual (UM08035) in a .PDF application.
Flasher Hub User Guide	Opens the Flasher Hub Manual (UM08039) in a .PDF application.
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.

Chapter 4

Settings

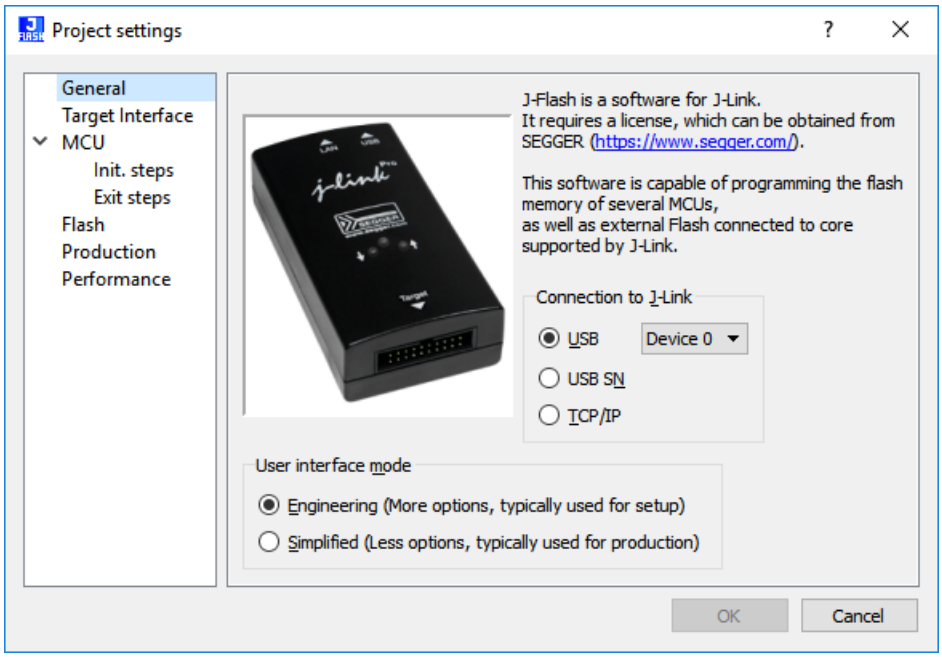
The following chapter provides an overview of the program settings. Both general and per project settings are considered.

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 configure the connection to the J-Link / Flasher.
The J-Link / Flasher can either be connected to the host system of J-Flash directly via USB, Ethernet or WiFi, or it can be connected through the J-Link Remote Server running on a remote system.
For more information on the operation of J-Link / Flasher, please refer to the J-Link Manual (UM08001) . For more information on the J-Link Remote Server, please refer to: (*J-Link Remote Server overview*) .



General Settings

USB

If this option is checked, J-Flash will connect to J-Link / Flasher over the USB port. The default device number is 0. For more information about how to use multiple J-Link / Flasher on one PC, please see also the chapter "Working with J-Link" of the J-Link Manual (UM08001).

USB S/N

If this option is checked, J-Flash will connect to J-Link / Flasher over the USB port. J-Flash will only use the J-Link / Flasher with the specified S/N and any operation will fail if the J-Link / Flasher with specified S/N is not connected or cannot be used for any reason.

TCP/IP

If this option is checked, J-Flash will connect to J-Link / Flasher via TCP/IP. This should be configured for SEGGER probes connected via Ethernet directly (e.g. J-Link PRO, Flasher PRO, …) or using the J-Link Remote Server. The hostname (or IP address) of the system to connect to be may be entered in the textbox for TCP/IP. For connections via the J-Link Remote Server the according connection string may be entered.

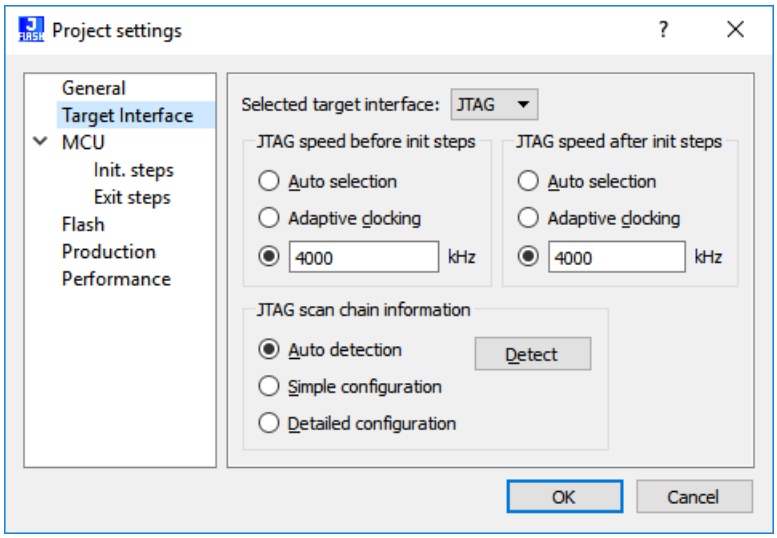
User interface mode

Select the *Engineering* radio button when setting up a project or
select the *Simplified* radio button when using J-Flash in production environments.

In the simplified user interface some options are disabled to reduce possible error sources in the production phase.

4.1.2 Target Interface Settings

This dialog is used to configure the interface connection to the target.



Target Interface Settings

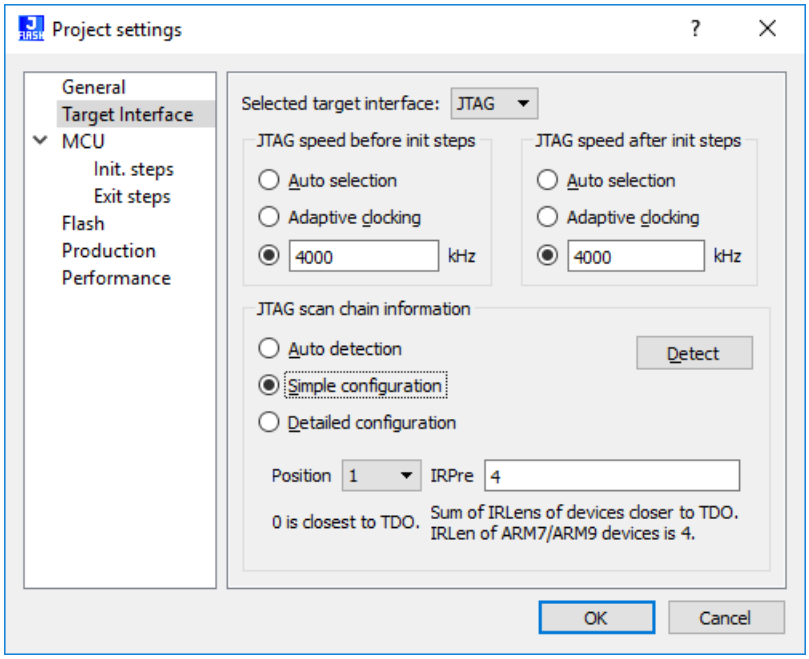
4.1.2.1 Interface Speed

The interface speed used before and after initialization can be configured. The interface speed before init is used to communicate with the target before and during execution of the custom initialization sequence (described in the section *Init steps* on page 33). The interface speed after init is used to communicate after executing the custom initialization sequence. This is useful if a target running at slow speed and the users wants to set up a PLL in the initialization sequence.

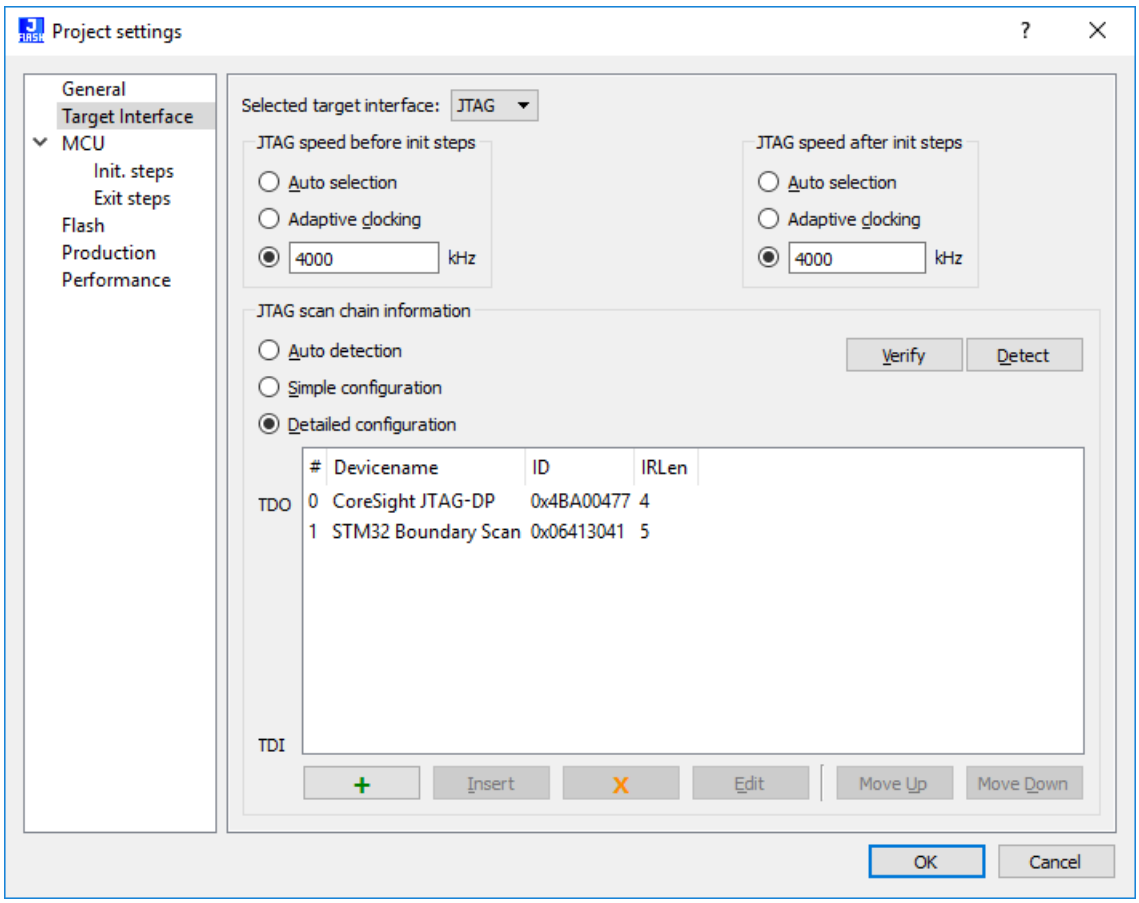
For more information about the different types of interface speed please see the chapter “Setup” of the J-Link Manual (UM08001).

4.1.2.2 JTAG scan chain

The “JTAG scan chain information” box allows to configure a JTAG scan chain with multiple devices on it.

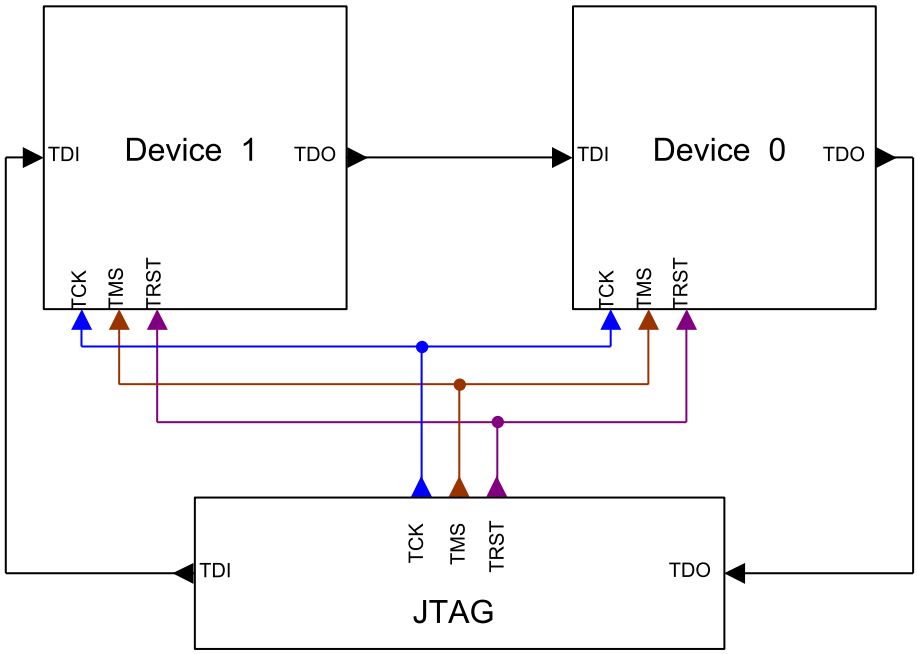


A "simple" JTAG scan chain configuration



A "detailed" JTAG scan chain configuration

In a scan chain configuration with multiple devices, the TCK and TMS lines of all JTAG devices are connected, while the TDI and TDO lines form a ring.

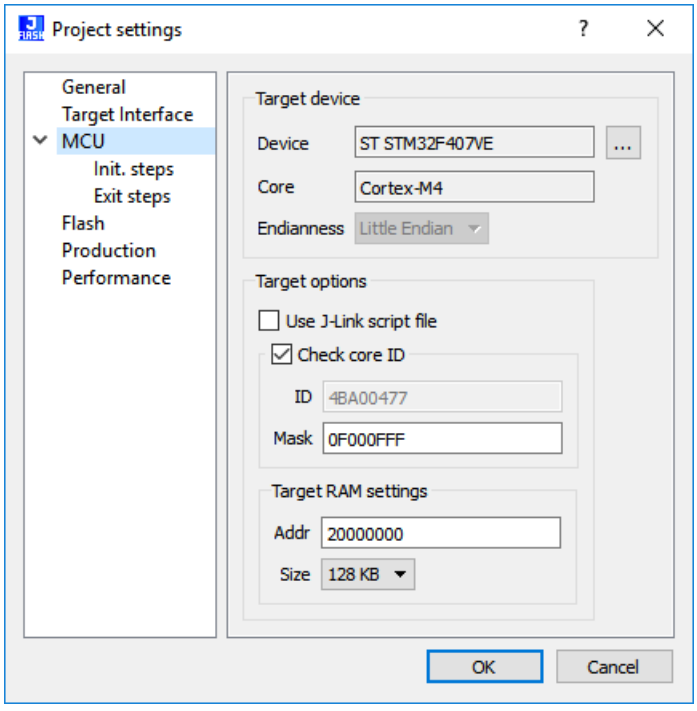


JTAG-chain

The position of the device to connect with J-Flash is selected from the Position dropdown 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 MCU Settings

This dialog allows the selection of microcontroller dependent settings.



MCU 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, the respective microcontroller must be selected in the **Device** list by clicking the **...** button. If a microcontroller is not found on this list, contact SEGGER, as new microcontrollers are continuously being added.

Device

Select the respective microcontroller from the list to program internal flash devices. In order to program external flash, select the device or core from the list.

Clock speed

For some devices, the actual CPU clock frequency in Hz of the MCU is required to guarantee accurate operation of J-Flash. By default, J-Flash uses the "Auto" CPU speed detection feature.

Endianness

The compatible endianness of the selected device is set automatically if possible. Otherwise, select **little endian** or **big endian** from the dropdown menu accordant to the device.

4.1.3.1 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 with the J-Link / Flasher is the intended device. The core ID for all listed devices is known, therefore this value is filled in automatically if a device is selected and can not be modified. If only a core family is selected, the core **ID** field can be modified.

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 the J-Link / Flasher 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.
}
```

4.1.3.2 Target RAM settings

The target RAM is used during flash programming to store the RAMCode. This RAM is called work RAM. The "Target RAM settings" define the start address and the size size of the work RAM.

4.1.3.3 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 the user must first perform the required initialization.

This dialog allows the user to enter a custom initialization sequence using a predefined 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 Manual (UM08001) for an detailed explanation of the different reset types.
Read 8bit	Address (Hex)	--	Reads 8bit from a given address and stores the value in the internal variable.
Read 16bit	Address (Hex)	--	Reads 16bit from a given address and stores the value in the internal variable.
Read 32bit	Address (Hex)	--	Reads 32bit from a given address and stores the value in the internal variable.
SetAllowRemoteRead	--	On/Off	This option defines if the emulator (remote) or the host handles the read access to the target. This option is activated by default to enhance the performance.
SetAllowRemoteWrite	--	On/Off	This option defines if the emulator (remote) or the host handles the write access to the target. This option is activated by default to enhance the performance.
Verify 8bit	Address (Hex)	Data (Hex)	Verifies whether 8bit data on a declared address is identical to the declared 8bit data. Verification failure is handled as error and causes an abort.
Verify 16bit	Address (Hex)	Data (Hex)	Verifies whether 16bit data on a declared address is identical to the declared 16bit data. Verification failure is handled as error and causes an abort.

Command	Value0	Value1	Description
Verify 32bit	Address (Hex)	Data (Hex)	Verifies whether 32bit data on a declared address is identical to the declared 32bit data. Verification failure is handled as error and causes an abort.
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&Verify 8bit	Address (Hex)	Data (Hex)	Writes 8bit data to a given address and verifies it afterwards. Verification failure is handled as error and causes an abort.
Write&Verify 16bit	Address (Hex)	Data (Hex)	Writes 16bit data to a given address and verifies it afterwards. Verification failure is handled as error and causes an abort.
Write&Verify 32bit	Address (Hex)	Data (Hex)	Writes 32bit data to a given address and verifies it afterwards. Verification failure is handled as error and causes an abort.
Write Register	Register	Data (Hex)	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.
SetModeBigEndian	--	--	Sets bit 7 of the CP15 register to 1.
SetModeLittleEndian	--	--	Sets bit 7 of the CP15 register to 0.

Command	Value0	Value1	Description
Var Write File 8bit	Address (Hex)	--	Writes 8bit data of the internal variable to a given address in the data file.
Var Write File 16bit	Address (Hex)	--	Writes 16bit data of the internal variable to a given address in the data file.
Var Write File 32bit	Address (Hex)	--	Writes 32bit data of the internal variable to a given address in the data file.
Comment	--	--	Can be used as additional space to insert comments. Does nothing.
Write File 8bit	Address (Hex)	Data (Hex)	Writes 8bit data to a given address in the data file.
Write File 16bit	Address (Hex)	Data (Hex)	Writes 16bit data to a given address in the data file.
Write File 32bit	Address (Hex)	Data (Hex)	Writes 32bit data to a given address in the data file.
Report Error	Value (Hex)	--	Displays a message box in J-Flash with the given error code.

Note

All “Write *” commands may only be used to write RAM or SFR registers, but not Flash memory. Flash memory can only be influenced by altering the data file. The data file can changed in the init steps by using the “Write File*” commands.

4.1.3.4 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**), those steps will be performed after verify.

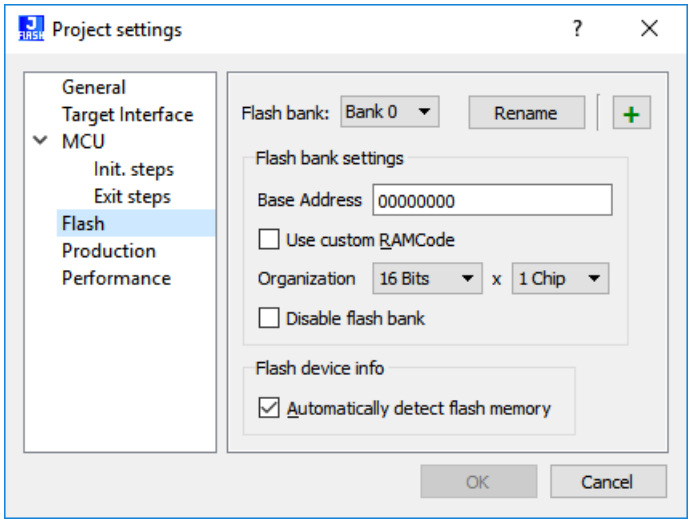
The Exit steps can be used to do some special handling after programming, for example to set some security bits in order to secure the chip.

Note

Exit steps are only performed for **Target** -> **Production Programming** operations.

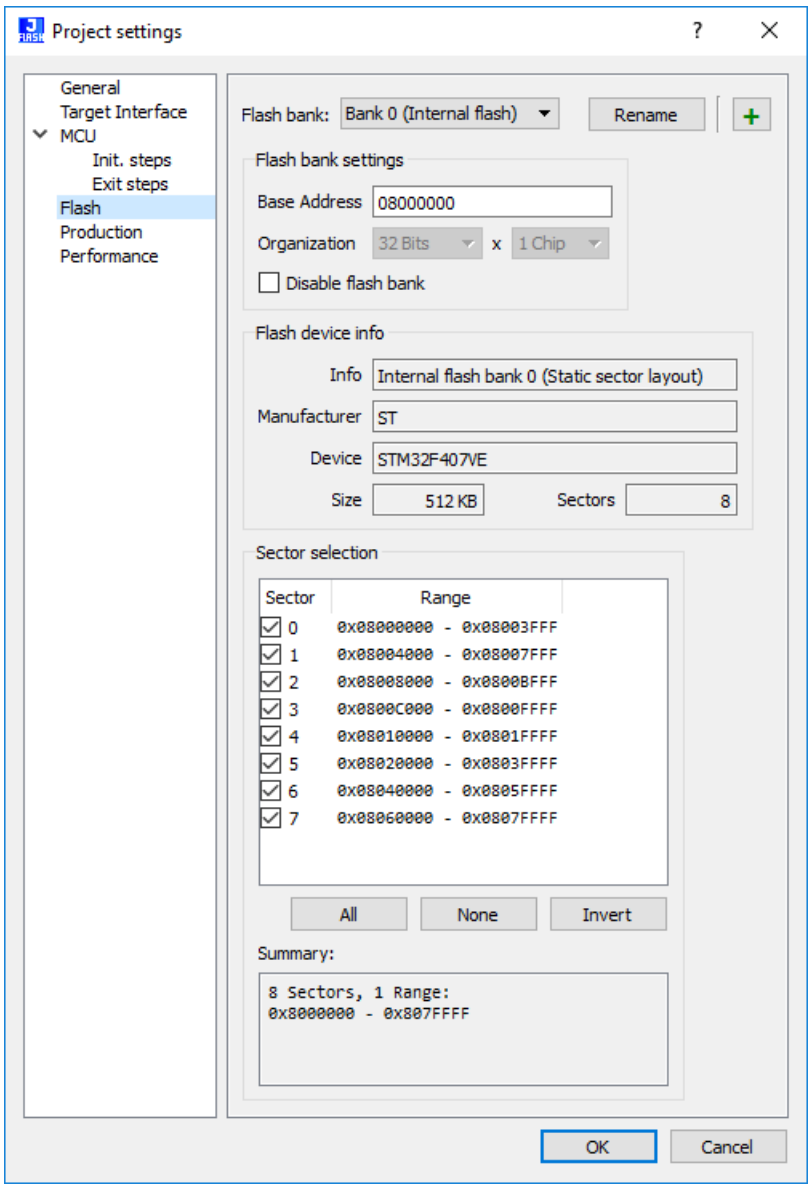
4.1.4 Flash Settings

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 *MCU Settings* on page 32. If a core family has been selected in order to program external flash memory or a custom Flash Bank has been added and is selected, the menu should look similar to the screenshot below.



Flash Settings: external Flash

If a specific device has been selected to program the flash of these device, the menu should look similar to the screenshot below.



Flash Settings: internal Flash

4.1.4.1 Base Address

This is the base address of the flash.

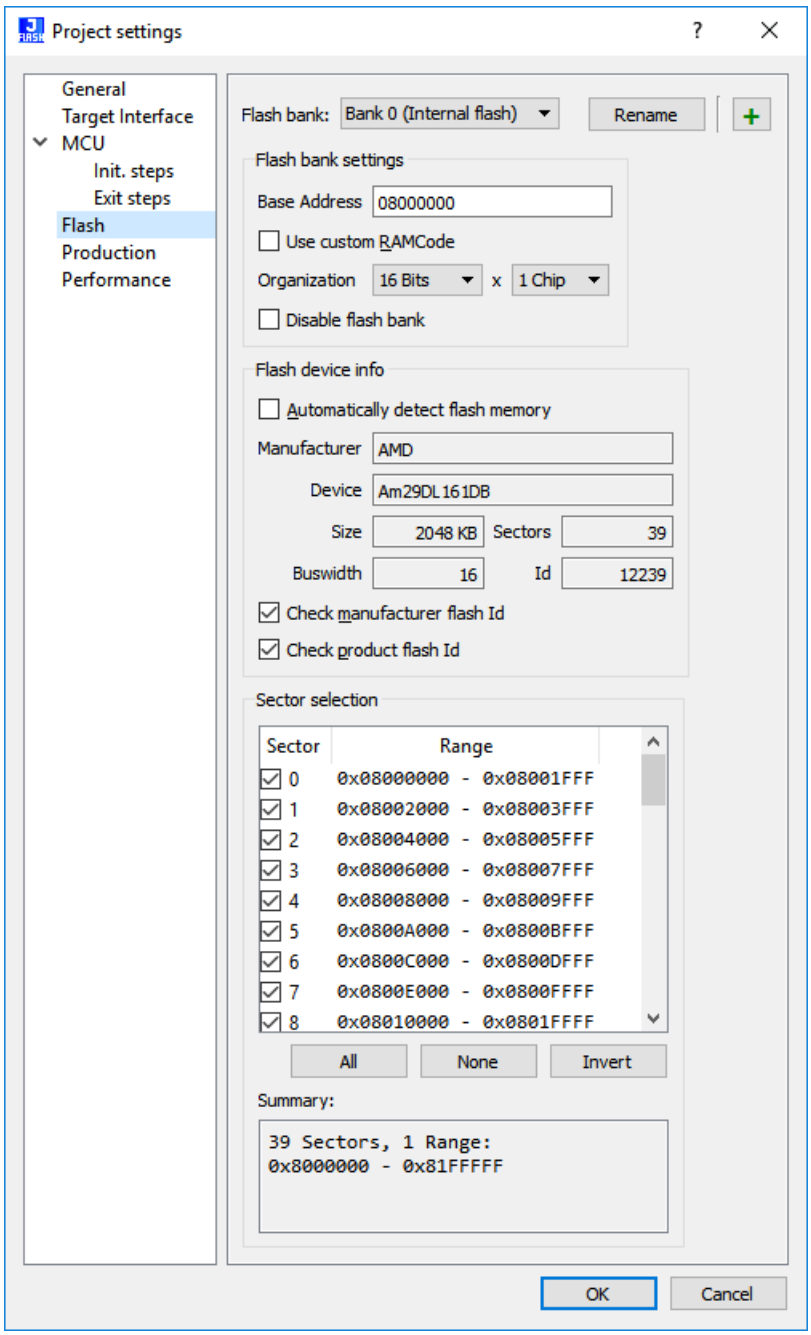
4.1.4.2 Disable flash bank

The **Disable flash bank** checkbox disables the flash bank currently selected in the drop-down menu, which can be used to change which flash banks are processed by J-Flash with only one click per flash bank. This is especially useful when testing different configurations.

4.1.4.3 Sector selection

The final section of this dialog indicates the sectors to be affected by erase, read and write operations done by J-Flash. An individual or series of sectors may be selected from the predetermined valid range.

4.1.4.4 External Flash specific settings



Flash Settings: external Flash, auto detection unchecked

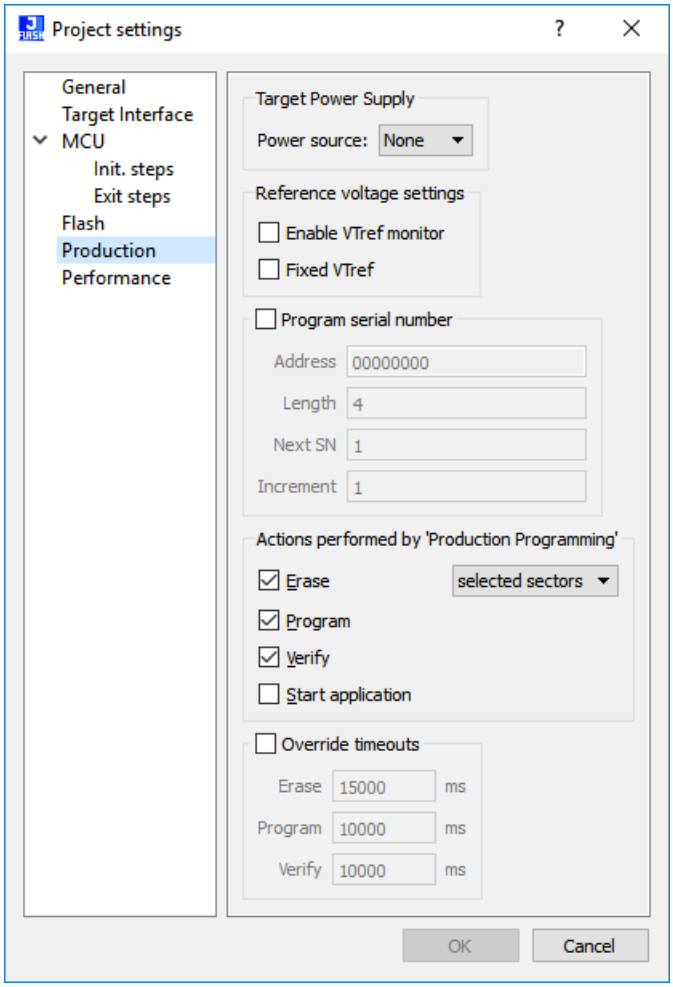
Organization

For some flashes (e.g. CFI compliant NOR flashes), the organization needs to be specified. The organization settings make it possible to configure the bus width and the number of flash chips connected to the address and data bus of the MCU.

ID checking

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

4.1.5 Production settings



Production Settings

Target power supply

Available power source options are:

- None
- V_{CC5V}
- V_{TGT}

Delay before start defines the delay (in ms) after enabling the target power supply and before starting to communicate with the target.
Discharge target on disconnect causes a discharge of any capacities left on the target on disconnect.

> **Note**

The option **V_{TGT}** for power source is for Flasher ATE only. Other Flasher models will use **V_{CC5V}**, even when **V_{TGT}** is selected.

Note

The option **Discharge target on disconnect** is for Flasher ATE only and will be ignored by the other Flasher models.

Reference voltage settings

Enabling **VTref monitor** causes the Flasher to monitor the target voltage (V_{Tref}) in standalone mode and makes the Flasher throw an error when the voltage drops below the minimum or rises above the maximum during programming.

Program serial number

J-Flash supports programming of serial numbers into the target in two ways. For a detailed description on how to use the serial number programming feature please refer to *Serial number programming* on page 21

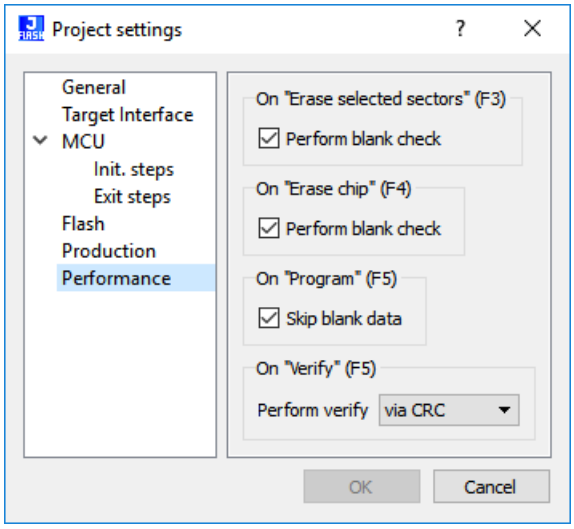
Actions performed by "Production Programming"

The checked options will be performed when auto programming a target via **Target -> Production Programming** (shortcut: F7)

Find below a table which describes the commands:

Command	Enabled by default?	Description
Init steps	Yes	Executes the init steps defined in the MCU settings
Erase sectors	Yes	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	Yes	Programs the data file.
Verify	Yes	Verifies the program data. • CRC: Verifies data via a high optimized CRC calculation (recommended verification method). • Complete data: Verifies data by reading it back.
Start application	No	Starts application after programming/verify completed. Needs reset pin to be connected to Flasher.
Secure chip	No	Secures the device if supported by algorithm.
Exit steps	Yes	Executes the exit steps defined in the MCU settings

4.1.6 Performance settings



Performance Settings

On "Erase selected sectors" / On "Erase chip"

If **Perform blank check** is checked, a blank check will be performed before an erase. If the area to erase is already blank, no erase happens.

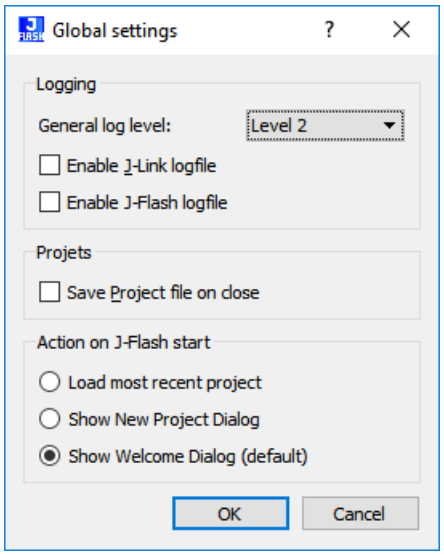
> **Note**
>
> This check takes place for **Target -> Production Programming** as well as **Target -> Manual Programming.**

On "Verify"

Configures the type of verify that takes place:

- via CRC
- via readback

4.2 Global Settings



Global Settings

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

General log level

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

Enable J-Link logfile

If this option is checked, a file name of the J-Link logfile can be specified. 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 DLL API functions called from within J-Flash.

Enable J-Flash logfile

If this option is checked, a file name of the J-Flash logfile can be specified. The J-Flash logfile contains the same messages as the log window output of J-Flash.

Save project file on close

If this option is checked, J-Flash will always save the changes made to a project file when a project or J-Flash is closed and therefore overrides the old project file without asking for permission to do so.

Action on J-Flash start

In this section, the action J-Flash performs on startup can be selected. *Load most recent project *Show New Project Dialog *Show Welcome Dialog (default)

Chapter 5

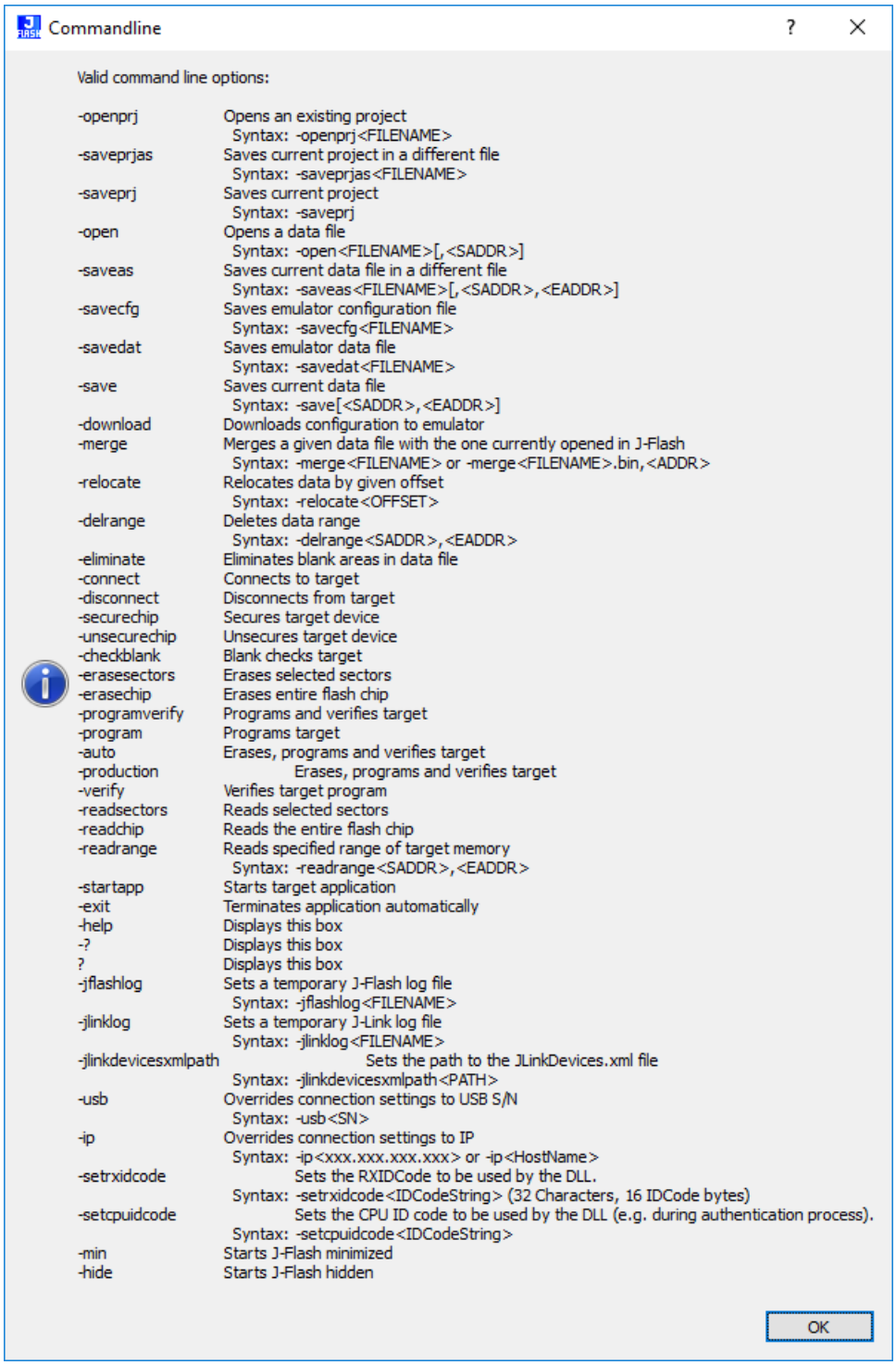
Command Line Interface

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

5.1 Overview

In addition to its graphical user interface (GUI), J-Flash supports a command line mode as well. This makes it possible to use J-Flash for batch processing or automatization purposes. All important options accessible from the menus are available in command line mode as well. If command line options are provided, 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 J-Flash is started with the command line parameter ***-help*** or ***-?***:
JFlash.exe -help
JFlash.exe -?



Command line options

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. <FILENAME>.
If these parameters are optional they are enclosed in square brackets too, e.g. [<SADDR>]. Neither the angle nor the square brackets must be typed on the command line, they are used here only to denote (optional) parameters.

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 occurred.

Option	Description
-?	Displays the help dialog.
-auto	Executes the steps selected in Production Programming. Default: Erases, programs and verifies target.
-checkblank	Blank checks target.
-connect	Connects to the target.
-delrange<SADDR>,<EADDR>	Deletes data in the given range.
-disconnect	Disconnects from the target.
-download	Downloads configuration to emulator.
-eliminate	Eliminates blank areas in data file.
-erasechip	Erases the entire flash chip.
-erasesectors	Erases selected sectors.
-exit	Exits J-Flash.
-help	Displays the help dialog.
-jflashlog<FILENAME>	Sets a temporary J-Flash logfile.
-jlinklog<FILENAME>	Sets a temporary J-Link logfile.
-jlinkdevicesxmlpath<PATH>	Sets the path to the JLinkDevices.xml file.
• -merge<FILENAME> • -merge<FILENAME>.bin,<ADDR>	Merges a given data file with the one currently opened in J-Flash.
-min	Starts J-Flash minimized
-open<FILENAME>[,<SADDR>]	Opens a data file
-openprj<FILENAME>	Opens an existing project file. This will also automatically open the data file that has been recently used with this project.
-production	Same as -auto
-program	Programs the target.
-programverify	Programs and verify the target.
-readchip	Reads the entire flash chip.
-readsectors	Reads selected sectors.
-readrange<SADDR>,<EADDR>	Reads specified range of target memory.
-relocate<OFFSET>	Relocates data by the given offset.

Option	Description
-save[<SADDR>,<EADDR>]	Saves the current data file. Please note that the parameters <SADDR>,<EADDR> apply only if the data file is a *.bin file or *.c file.
-saveas<FILENAME>[,<SADDR>,<EADDR>]	Saves the current data file in 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	Saves the current project.
-saveprjas<FILENAME>	Saves the current project in the specified file.
-securechip	Secures target device.
-setcpuidcode<IDCodeString>	Sets the CPU ID code to be used by the DLL (e.g. during authentication process).
-setrxidcode<IDCodeString>	Sets the RXIDCode to be used by the DLL. Additional info <IDCodeString>: 32 characters, 16 IDCode bytes
-startapp	Starts the target application.
-unsecurechip	Unsecures target device.
-verify	Verifies the target memory.
-usb<SN>	Overrides connection settings to USB S/N.
• -ip<xxx.xxx.xxx.xxx> • -ip<HostName>	Overrides connection settings to IP.
-hide	Starts J-Flash hidden.

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 command line options are provided, 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 ("Production Programming")
4. Close J-Flash

The return value will be checked and in case of an error an error message displayed. This sample can be used as a template to be adapted 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 minimized

The following example call starts J-Flash minimized:

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

> **Note**
>
> 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"*
- A J-Flash project (containing the configuration).

Basically, J-Flash connects to a specific J-Link / Flasher, 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.

5.4.1 Batch file example

Find below a small example which shows how to program multiple targets in parallel using a batch file (Windows only). Please note that in this example, two separate files are required to start parallel programming with a batch file.

ParallelProgramming.bat

Executing this file starts parallel programming using J-Flash.

> **Note**
>
> The "`List of jobs`" section contains placeholders and must be adjusted to use the desired Flasher serial numbers, J-Flash project file(s) and data file(s). Furthermore, the list may be expanded to run more sessions of J-Flash in parallel.

```
@ECHO OFF
REM
REM List of jobs
REM          Serial no.    J-Flash project file         Data file
set aJobs[0]=1015000001 Path\To\ProjectFile.jflash Path\To\DataFile.hex
set aJobs[1]=1015000002 Path\To\ProjectFile.jflash Path\To\DataFile.hex
set aJobs[2]=1015000003 Path\To\ProjectFile.jflash Path\To\DataFile.hex
REM set aJobs[3]=1015000004 Path\To\ProjectFile.jflash Path\To\DataFile.hex
REM set aJobs[4]=1015000005 Path\To\ProjectFile.jflash Path\To\DataFile.hex
REM set aJobs[5]=1015000006 Path\To\ProjectFile.jflash Path\To\DataFile.hex
REM set aJobs[6]=1015000007 Path\To\ProjectFile.jflash Path\To\DataFile.hex
REM [...]
REM

REM
REM   :Main
REM
REM  Function description
REM    Entry point for the batch script
REM    Starts multiple instances of J-Flash and waits until all of them have
 exited
REM
:Main
  REM
  REM Enable the use of variables inside the for loop by using delayed variable
 expansion
  REM
  setlocal ENABLEDELAYEDEXPANSION
  REM
  REM In order to wait for all processes to finish, lock files are used which are
 located at %temp%
  REM Each process blocks its corresponding lock file as long as the process is
 alive.
  REM
  set  "lock=%temp%\wait!random!.lock"
```



```
  echo Starting J-Flash...
  set /a Cnt=0
  :_JobStartLoop
  if defined aJobs[%Cnt%] (
    start "" 9>"!lock!%Cnt%" StartJFlash.bat %%aJobs[%Cnt%]%%
    set /a "Cnt+=1"
    GOTO :_JobStartLoop
  )
  echo Waiting for J-Flash to finish...
  REM
  REM Wait for processes to finish before continuing
  REM
  set /a Cnt=0
  :_JobWaitLoop
  if defined aJobs[%Cnt%] (
    call :WaitForUnlock !lock!%Cnt% >nul 2>&1
    set /a "Cnt+=1"
    GOTO :_JobWaitLoop
  )
  REM
  REM Delete temporary lock files
  REM
  del "!lock!*"
  echo Done.
  pause
  exit /b

REM
REM   :WaitForUnlock
REM
REM  Function description
REM    This function waits for the passed lock file to be accessible
REM
REM  Parameters
REM    %~1    Lock file path
REM
:WaitForUnlock
  goto :Start
  :Retry
  REM
  REM This is a ping to the IPv6 local loopback address which is used to burn
 some time waiting for J-Flash to finish.
  REM There is a 1 sec delay between two pings, so /n 2 generates a sleep for at
 least 1 sec.
  REM
  1>nul 2>nul ping /n 2 ::1
  :Start
  call 9>"%~1" || goto Retry
  exit /b
```

StartJFlash.bat

This file is used as a helper file for `ParallelProgramming.bat` and should not be executed on its own.

> **Note**
>
> The path to the J-Flash executable ("`JFlash.exe`") may be adjusted to fit the environment this file is used in.

```
@ECHO OFF
REM
REM Expected parameters passed to this script:
REM   %1    S/N of USB Flasher
REM   %2    Path to project file
```



```
REM   %3    Path to data file
REM
REM Open a project with a data file, start programming and exit afterwards
REM
start /wait "J-Flash" "JFlash.exe" -usb%1 -openprj%2 -open%3 -auto -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
```

5.4.2 Python script example

Find below a small example which shows how to program multiple targets in parallel using Python 3.

ParallelProgramming.py

> **Note**
>
> The list of job information (_aJobs) contains placeholders and must be adjusted to use the desired J-Flash command line options. Furthermore, the list may be expanded to run more sessions of J-Flash in parallel. Additionally, the path to the J-Flash executable (_PATH_JFLASH_EXE) may be adjusted to fit the environment this file is used in.

```
#/*********************************************************************
#*
#*       Version Check
#*
#*********************************************************************
#*/
import os
import sys
import subprocess

if sys.version_info < (3, 0, 0):
  sys.stderr.write("Warning: Using Python 2.x. This script may only be run using
 Python 3!\n")
  raw_input("Press <Return> to exit")
  sys.exit(1)

#/*********************************************************************
#*
#*       Defines
#*
#*********************************************************************
#*/

_PATH_JFLASH_EXE = os.path.join("C:\\", "Program Files", "SEGGER", "JLink",
 "JFlash.exe")  # Default installation path on Windows

#/*********************************************************************
#*
#*       Classes
#*
#*********************************************************************
#*/

class JOB_INFO:
```



```
  def __init__(self, ParamList):
    self.ParamList = ParamList    # List of strings, each of which represents a
 command line option passed to J-Flash

#/*****************************************************************
#*
#*       List of jobs
#*
#*****************************************************************
#*/

_aJobs = [  # For a list of supported command line options, refer to https://
wiki.segger.com/UM08003_JFlash#Command_line_options
   JOB_INFO(["-usb1015000001", "-openprjPath\\To\\ProjectFile.jflash", "-openPath
\\To\\DataFile.hex", "-auto", "-exit"])
  ,JOB_INFO(["-usb1015000002", "-openprjPath\\To\\ProjectFile.jflash", "-openPath
\\To\\DataFile.hex", "-auto", "-exit"])
  ,JOB_INFO(["-usb1015000003", "-openprjPath\\To\\ProjectFile.jflash", "-openPath
\\To\\DataFile.hex", "-auto", "-exit"])
# ,JOB_INFO(["-usb1015000004", "-openprjPath\\To\\ProjectFile.jflash", "-openPath
\\To\\DataFile.hex", "-auto", "-exit"])
# ,JOB_INFO(["-usb1015000005", "-openprjPath\\To\\ProjectFile.jflash", "-openPath
\\To\\DataFile.hex", "-auto", "-exit"])
# ,JOB_INFO(["-usb1015000006", "-openprjPath\\To\\ProjectFile.jflash", "-openPath
\\To\\DataFile.hex", "-auto", "-exit"])
# ,JOB_INFO(["-usb1015000007", "-openprjPath\\To\\ProjectFile.jflash", "-openPath
\\To\\DataFile.hex", "-auto", "-exit"])
#  [...]
]

#/*****************************************************************
#*
#*       Functions / Code
#*
#*****************************************************************
#*/

#/*****************************************************************
#*
#*       main()
#*
#*  Function description
#*    Handles parallel programming via multiple J-Flash instances
#*/
def main():
  aProc = []
  #
  # Sanity check
  #
  r = os.path.exists(_PATH_JFLASH_EXE)
  if (r == False):
    print("Could not find J-Flash at", _PATH_JFLASH_EXE)
    input("Press <Return> to exit...")
    return -1
  #
  # Go through jobs and start J-Flash instance for each job
  #
  print("Starting J-Flash...")
  for JobInfo in _aJobs:
    aCmd = [_PATH_JFLASH_EXE]  # The first arg passed to subprocess.Popen is the
 path to the executable
    aCmd += JobInfo.ParamList
    hProc = subprocess.Popen(aCmd, shell=True)
    aProc.append(hProc)
  #
  # Go through processes and wait for each to finish executing
  #
  print("Waiting for J-Flash to finish...")
```



```
  Result = 0
  for hProc in aProc:
    hProc.wait()
    r = hProc.returncode
    if (r != 0):          # Error occurred? => Print error message including
 arguments for identification of failed session
      Result = -1
      print("Error occurred for following J-Flash instance:", hProc.args)
  if (Result == 0):  # No errors occurred?
    print("Success!")
  print("Done.")
  input("Press <Return> to exit...")
  return Result

#/*************************************************************************
#*
#*       Main
#*
#*************************************************************************
#*/
if __name__ == "__main__": # only executed if this module is executed directly,
 not if its imported
  main()
```


Chapter 6

Device specifics

For some devices, special handling might be required. In order to find out if special handling is necessary for the used device, please refer to the SEGGER wiki: *SEGGER Wiki: Device specifics*

Chapter 7

Target systems

The following chapter lists all supported flash devices.

7.1 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, the flash device has to explicitly selected 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 56.

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.
For more information on how to create a custom RAM Code for J-Flash, please refer to the SEGGER Wiki: *SEGGER Wiki: Creating a Flash Loader*

For more information about which which microcontrollers with internal flash are supported by J-Flash, please refer to *Supported microcontrollers* on page 56.

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

7.2 Supported microcontrollers

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

List of supported devices

7.2.1 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. The latest list of supported flash devices can always be found on our website:

List of supported flash devices

Chapter 8

Performance

For programming speed measurements, please refer to the SEGGER website: *Internal Flash programming performance*

Chapter 9

Background information

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

9.1 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.

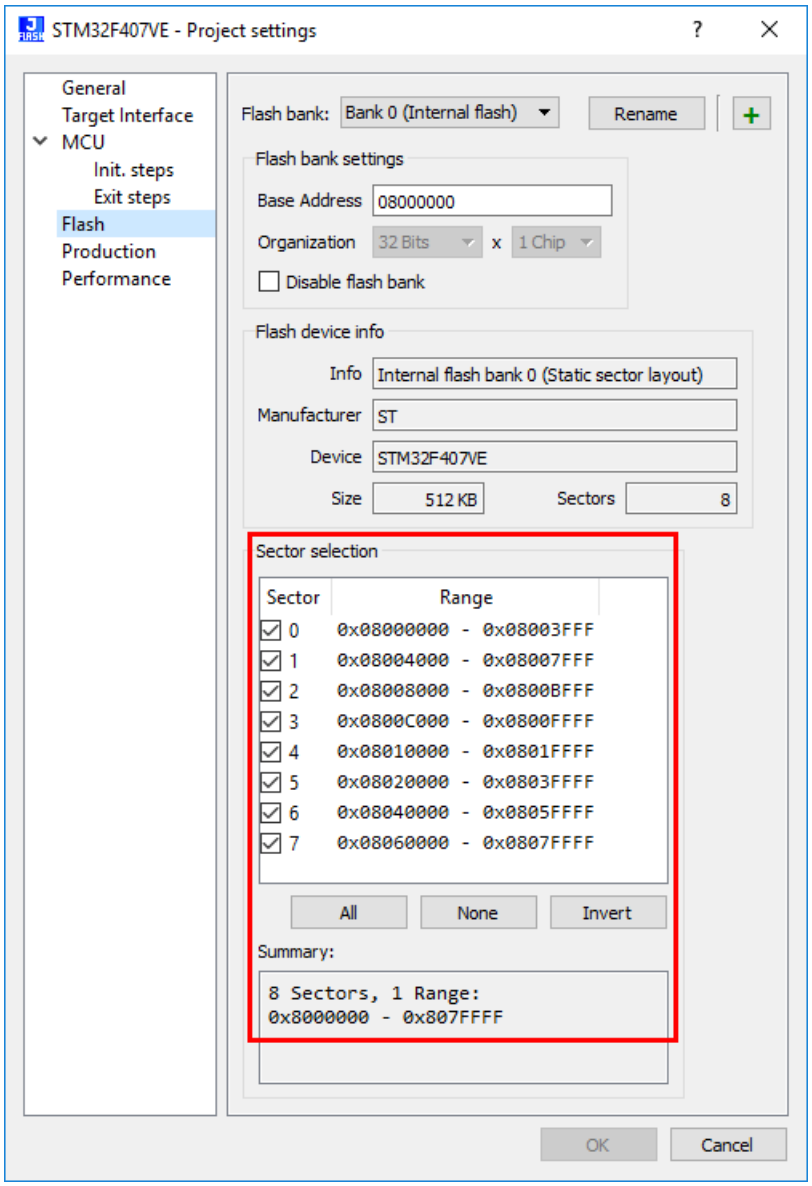
Log

```
Application log started
 - J-Flash V6.71a (J-Flash compiled Apr  3 2020 11:59:16)
 - JLinkARM.dll V6.71a (DLL compiled Apr  3 2020 11:58:51)
Reading flash device list [C:\Program Files (x86)\SEGGER\JLink\ETC/JFlash/Flash.csv] ...
 - List of flash devices read successfully (451 Devices)
Reading MCU device list ...
 - List of MCU devices read successfully (7690 Devices)
Opening project file [C:\Work\STM32F407VE.jflash] ...
 - Project opened successfully
Opening data file [C:/Work/TestData.hex] ...
 - Data file opened successfully (524288 bytes, 1 range, CRC of data = 0x57679A97, CRC of file = 0xA88CE8D0)
```

CRC in Log Window

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



CRC: selected sectors

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.
2. The polynomial which is used for the CRC calculation is `0xEDB88320`.

Chapter 10

Support

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

10.1 Troubleshooting

10.1.1 General procedure

- Make sure the J-Link / Flasher is working as expected. See the troubleshooting article in the SEGGER wiki: *SEGGER Wiki: J-Link / J-Trace / Flasher Troubleshooting*
- 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
- The interface 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.

10.1.2 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 MCU Settings.

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 MCU Settings 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.

10.2 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 logfile and project file. In order to generate an expressive logfile, set the log level to “All messages” (see section Global Settings 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

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

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

