Flasher Secure

Secured Production at
Contract Manufacturers

User Guide & Reference Manual

Document: UM08032
Software Version: 1.0
Revision: 1
Date: May 8, 2018

A product of SEGGER Microcontroller GmbH

www.segger.com

Contact address

SEGGER Microcontroller GmbH

In den Weiden 11
D-40721 Hilden

Germany

Tel.	+49 2103-2878-0
Fax.	+49 2103-2878-28
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 inform 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: May 8, 2018

Software	Revision	Date	By	Description
1.00	0	170401	AB	Initial release.
1.00	1	180504	AB/ MF	Added FlasherSecureServer feature
1.00	2	180507	AB/ MF	Minor corrections after review

About this document

Assumptions

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

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

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

How to use this manual

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

Typographic conventions for syntax

This manual uses the following typographic conventions:

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

Table of contents

Chapter 1

Introduction

This chapter gives a short overview about the Flasher Secure.

1.1 Flasher Secure overview

Flasher is a programming tool for micro controllers with on-chip or external flash memory. Flasher is designed for programming flash targets with the J-Flash software or stand-alone. In addition to that, Flasher Secure can also be used as a regular J-Link. For more information about J-Link in general, please refer to the *J-Link / J-Trace User Guide* on page 90

In stand-alone mode, Flasher Secure can be driven by the start/stop button. Flasher Secure connects to a PC using the USB/Ethernet/RS232 interface. It can be controlled by handshake control or by ASCII command protocol. USB connection requires the J-Flash tool (for further detail take look into the *J-Flash manual* on page 90.

Flasher Secure always has a 20-pin connector. Many adapters are available. See chapter on page 74.

1.1.1 Features of Flasher Secure

- Three boot modes: J-Link mode, stand-alone mode, MSD mode
- Stand-alone JTAG, SWD and FINE programmer (Once set up, Flasher Secure can be controlled without the use of PC program).
- No power supply required, powered through USB
- Supports internal and external flash devices
- approximately 128 MB memory for storage of target program
- Can be used as J-Link (emulator)
- Data files may be updated via USB/Ethernet (using the J-Flash software), via FTP, via RS232 or via the MSD functionality of Flasher Secure

Flasher model	Supported cores	Supported target interfaces	Flash programming speed (depending on target hardware)
Flasher Secure	ARM7/ARM9/Cortex-M	JTAG, SWD	between 170 and 300 Kbytes/second

1.1.2 Working environment

General

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

Host System

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

Power supply

Flasher Secure: 5V DC, min. 100 mA via USB connector.

Host System Server

- IBM PC/AT or compatible CPU: 486 (or better) running at 1GHz or faster
- least 8 GB of RAM
- Operating System: Microsoft Windows 7, Windows 8, Windows 10 or Linux
- Ethernet interface available for communication with Flasher Secure.

Installing Flasher PC-software (J-Flash)

The latest version of the J-Flash software, which is part of the J-Link software and documentation package, can always be downloaded from our website:

https://www.segger.com/jlink-software.html For more information about using J-Flash please refer to `UM08003_JFlashARM.pdf` (J-Flash user guide) which is also available for download on our website.

1.2 Specifications

1.2.1 Specifications for Flasher Secure

General	
Supported OS	Microsoft Windows 7 Microsoft Windows 7 x64 Microsoft Windows 8 Microsoft Windows 8 x64 Microsoft Windows 10 Microsoft Windows 10 x64
Operating Temperature	+5 °C … +60 °C
Storage Temperature	-20 °C … +65 °C
Relative Humidity (non-condensing)	<90% rH
Mechanical	
Size (without cables)	121mm x 66mm x 30mm
Weight (without cables)	119g
Available interfaces	
USB Host interface	USB 2.0, full speed
Ethernet Host interface	10/100 MBit
RS232 Host interface	RS232 9-pin
Target interface	JTAG 20-pin (various adapters available)
Target Interface, Electrical	
Power Supply	USB powered, 100mA for Flasher Secure. 500 mA if target is powered by Flasher Secure
Target interface voltage (VIF)	1.2 … 5V
Target supply voltage	Supply voltage is 5V, max.
Target supply current	max. 400mA
Reset Type	Open drain. Can be pulled low or tristated
Reset low level output voltage (VOL)	VOL ≤ 10% of VIF
For the whole target voltage range (1.8V ≤ VIF ≤ 5V)	
LOW level input voltage (VIL)	VIL ≤ 40% of VIF
HIGH level input voltage (VIH)	VIH ≥ 60% of VIF
For 1.8V ≤ VIF ≤ 3.6V	
LOW level output voltage (VOL) with a load of 10 kOhm	VOL ≤ 10% of VIF
HIGH level output voltage (VOH) with a load of 10 kOhm	VOH ≥ 90% of VIF
For 3.6 ≤ VIF ≤ 5V	
LOW level output voltage (VOL) with a load of 10 kOhm	VOL ≤ 20% of VIF
HIGH level output voltage (VOH) with a load of 10 kOhm	VOH ≥ 80% of VIF
JTAG Interface, Timing	
Max. JTAG speed	up to 15MHz
Data input rise time (Trdi)	Trdi ≤ 20ns

Data input fall time (Tfdi)	Tfdi ≤ 20ns
Data output rise time (Trdo)	Trdo ≤ 10ns
Data output fall time (Tfdo)	Tfdo ≤ 10ns
Clock rise time (Trc)	Trc ≤ 10ns
Clock fall time (Tfc)	Tfc ≤ 10ns

1.2.1.1 Flasher Secure download speed

The following table lists the Flasher Secure performance values for writing to memory (RAM) via the JTAG interface:

Hardware	ARM memory download
Flasher Secure	~720 Kbytes/s (15MHz JTAG)

Note

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

1.2.1.2 Supported Target interfaces

The Flasher Secure supports the following target interfaces:

- JTAG
- SWD
- FINE
- SPD

1.2.2 Specifications for Flasher Secure Server

General	
Supported OS	Microsoft Windows 7 Microsoft Windows 7 x64 Microsoft Windows 8 Microsoft Windows 8 x64 Microsoft Windows 10 Microsoft Windows 10 x64 Linux

Chapter 2

The concept of the Flasher Secure

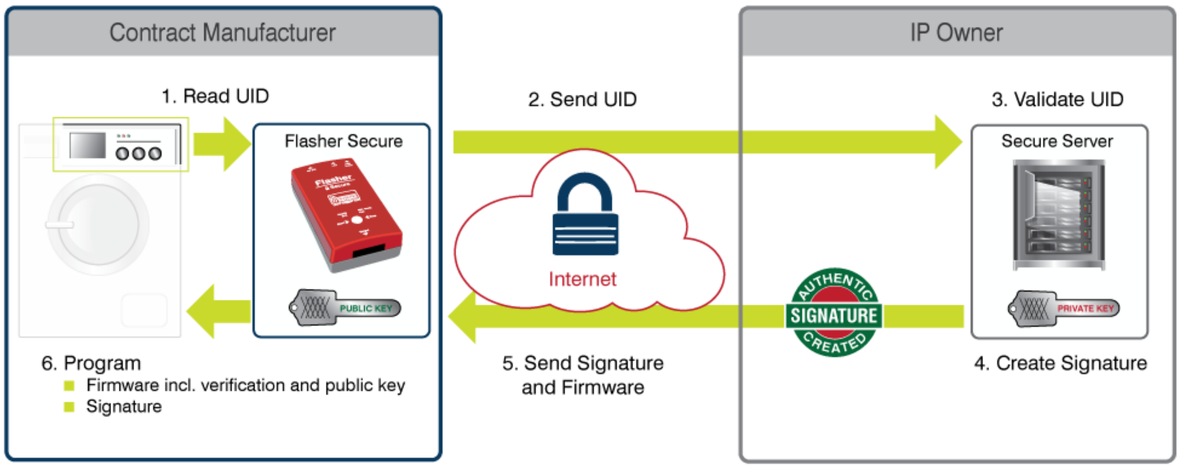
The Flasher Secure is a mass production tool, intended to be used at an external manufacturing company. The goal is to protect the intellectual property against unauthorized copying.

It consists of a server, which runs in a trusted environment. The server can either be hosted by SEGGER or by yourself. The second part is a client in the form of a Flasher device or a PC application. An IP connection via Internet between the server and the client is mandatory.

The following sub-chapters will explain the security concepts of the Flasher Secure solution.

2.1 Security based on emSecure

Most of today’s devices provide a “unique identifier” (UID). The UID is factory programmed by the chip vendor and cannot be altered. This characteristic in combination with asymmetric cryptography based on emSecure makes it possible to generate a digital signature unique for each device. The digital signature can be verified by the firmware running on the device, so using the firmware on another device will result in a signature failure.



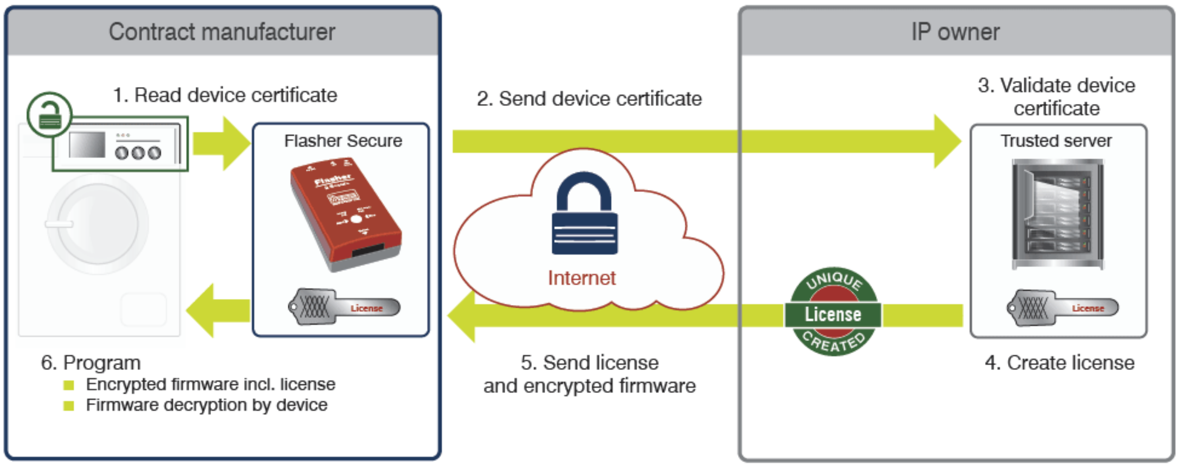
The Flasher Secure server supports emSecure-RSA and emSecure-ECDSA. While emSecure-ECDSA requires slightly less stack RAM compared to emSecure-RSA for verification at the same encryption strength, it significantly performs worse and requires more code space. So we recommend using emSecure-RSA, but it may make sense to use emSecure-ECDSA, if, for example, you are already using it for other purposes.

2.2 Third party solutions

Besides the emSecure digital signature solution, the Flasher Secure supports proprietary chip vendor specific solutions. These solutions provide end-to-end encryption: The firmware is decrypted on the target. If you use one of these solutions, there is no need for additional protection via digital signatures over the UID.

2.2.1 Secure firmware installation (ST)

The secure firmware installation process by ST uses a combination of symmetric and asymmetric cryptography. It uses a symmetrically encrypted firmware file, which is decrypted on chip. The chip contains a hidden asymmetric private key and a readable public key. The symmetric key for the decryption of the firmware is encrypted by the server with the chip's public key and sent to the chip together with the encrypted firmware data. The chip's public key additionally is signed by ST, so the server can determine if the public key originates from a real device.



Chapter 3

Flasher Secure server

The server, as the key component, controls the manufacturing process. It authorizes each produced device, keeps track of the manufacturers and intervenes if they behave in an unusual manner. And it also provides automatic firmware updates to the Flasher Secure clients.

The server provides an interface for the Flasher Secure clients and a web interface for administration. Any communication over the network is secured via transport layer security (TLS).

3.1 Installation

The Flasher Secure Server is available for the following operation systems:

- Windows and
- Linux.

3.1.1 Hardware Requirements

The following hardware is recommended for the Flasher Secure Server:

Hardware part	Required minimum
Cores	1
Core clock	1 GHz
RAM	8 GB
Harddrive	32GB (mainly depends on the number of projects and devices)

3.1.2 Connection Requirements

The Flasher Secure Server requires to be connectable at ports:

- port 3085 is used for the build in web server providing the user interface.
- port 3110 is used for the Flasher Secure unit to connect to the server.

3.1.3 Installation on a Linux system

The Linux installation requires only to follow a few steps:

- create a user account for the FlasherSecureServer, e.g. FlashSecServ
- create a directory for your instance of the Flasher Secure Server, e.g. `sudo mkdir /opt/SEGGER/FlasherSecureServer`
- change the owner of the directory, e.g. `sudo chown FlashSecServUsr /opt/SEGGER/FlasherSecureServer`, the Flasher Secure Server will put all its data into subdirectories of this root directory.
- copy the shipment `Flasher_Secure_Linux_x64.tar.gz` to the created folder
- extract the shipment in the folder, e.g. `tar xvzf Flasher_Secure_Linux_x64.tar.gz`
- create a startup script for your system, e.g. use the sample systemd config file to start the server binary with the correct user account.
- if you have firewall ensure that the FlasherSecureServer can be reached at the port 3085 and 3110.
- reboot your system or start the FlasherSecureServer application. (note: start the Flasher Secure Server from the installation directory and not with a fully qualified path from else where.)

Now everything is ready for the first use.

3.1.4 Installation on a Windows system

Available on request.

3.1.5 Server Certificates

You may generate the required certificates with OpenSSL. A script file for this task is provided in the CERT sub-directory. The server can be used with signed certificates as well as with self-signed certificates.

The RSA server certificate can be installed on the Flasher Secure Units (SERVER.CRT). If a certificate file is present on a Flasher Secure unit, the unit will only connect to a server using exactly this certificate effectively preventing man-in-the-middle attacks.

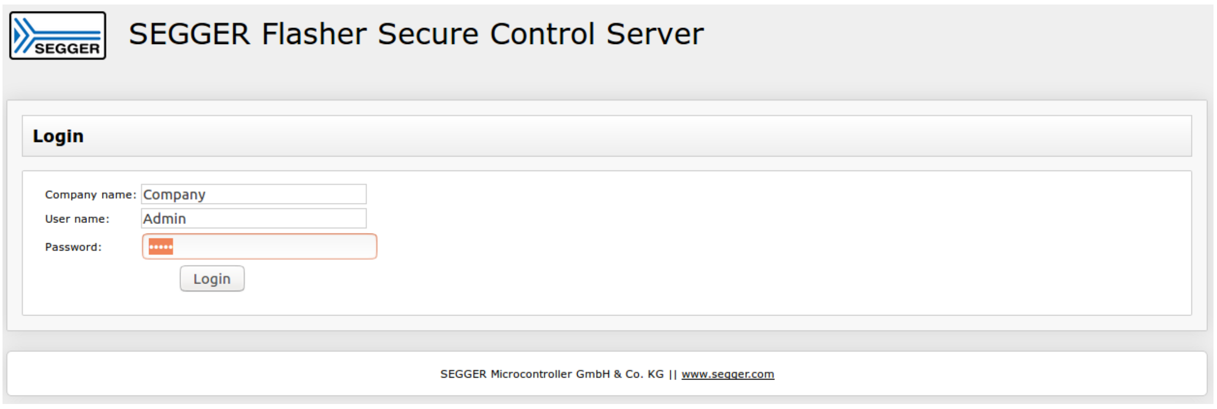
Note

The Flasher Secure units only support RSA. Therefore, the server must be provided at least an RSA certificate. The ECDSA certificate is optional, but can be used for accessing the web interface.

3.2 Web interface

3.2.1 Login

The Flasher Secure Server is reachable at `https://[server-ip]:3085`. The startpage is the login page:



Company name: Company
User name: Admin
Password: •••••

SEGGER Microcontroller GmbH & Co. KG || www.segger.com

The default logins are:

Company	User name	Password
Company	Admin	Admin
Company	User	User

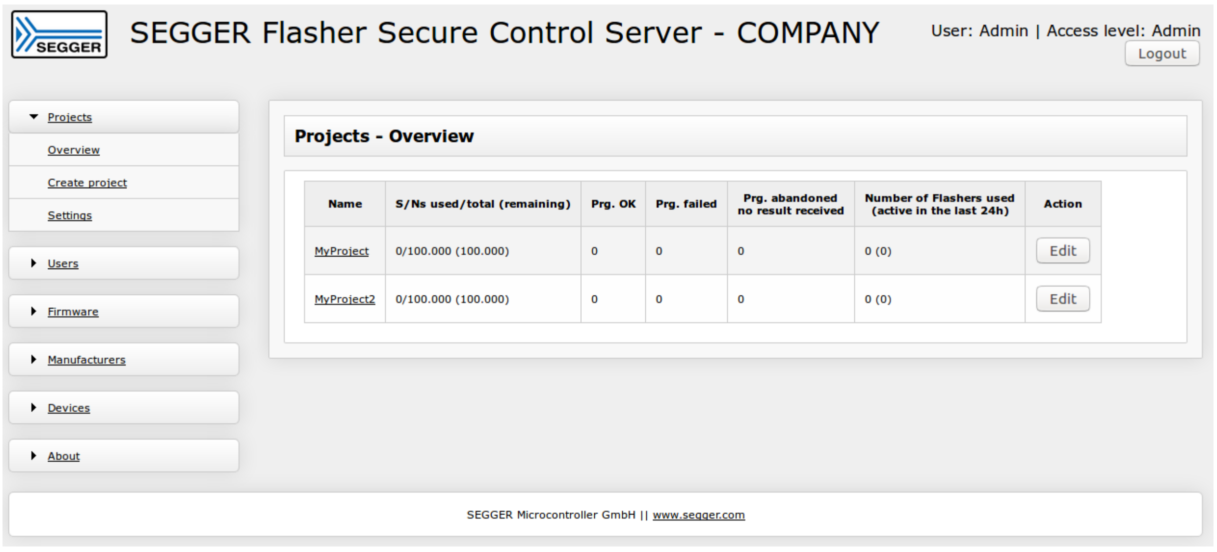
The Login levels are:

Login Level	privileges
Admininstrator	can read, write and change project and configuration settings.
User	can only read or view project and configuration settings.

The user administration is described in section *User configuration* on page 26.

3.2.2 Main page

The main page is the project overview, which shows information on the progress of each project.



SEGGER Flasher Secure Control Server - COMPANY

User: Admin | Access level: Admin

Projects - Overview

Name	S/Ns used/total (remaining)	Prg. OK	Prg. failed	Prg. abandoned no result received	Number of Flashers used (active in the last 24h)	Action
MyProject	0/100.000 (100.000)	0	0	0	0 (0)	Edit
MyProject2	0/100.000 (100.000)	0	0	0	0 (0)	Edit

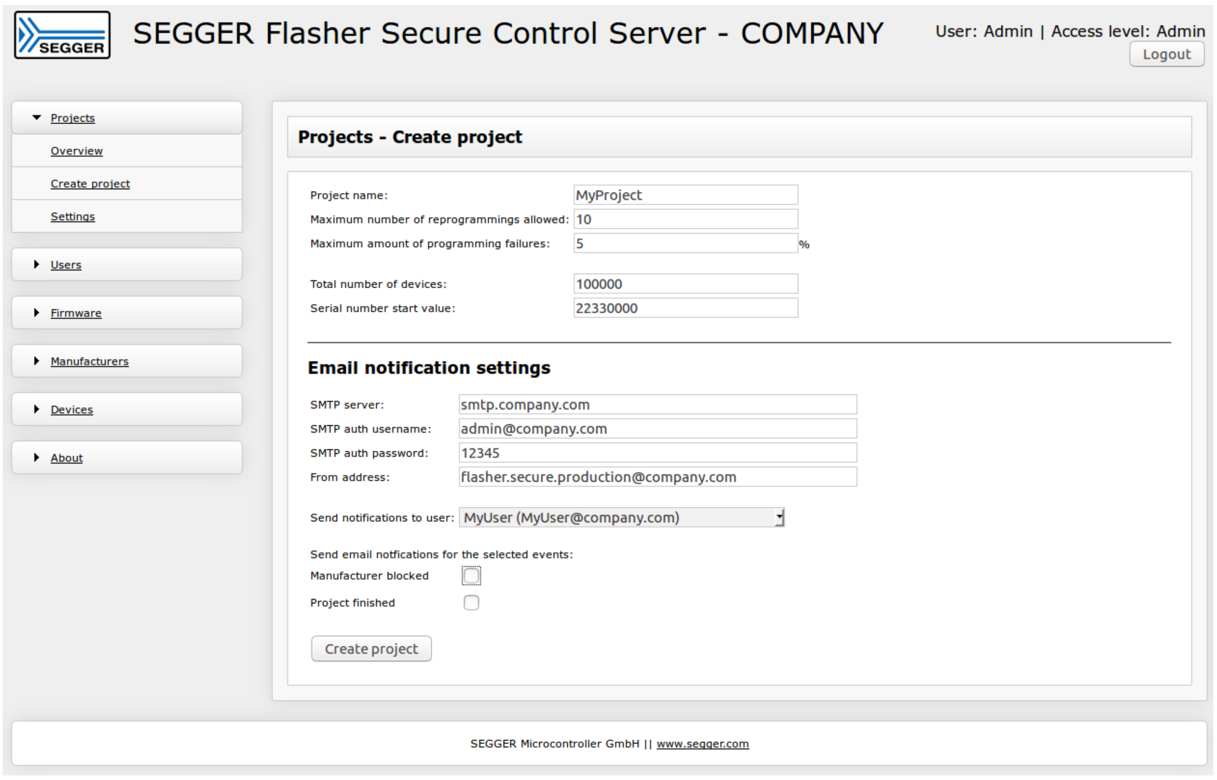
SEGGER Microcontroller GmbH || www.segger.com

The navigation is located on the left side. Every category has some subpages, for the provided features and actions.

3.2.3 Projects

3.2.3.1 Create a new project

The create a new project page creates a new project.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Overview
Create project
Settings
Users
Firmware
Manufacturers
Devices
About
Create project
SEGGER Microcontroller GmbH || www.segger.com

Projects - Create project

Project name:	MyProject
Maximum number of reprogrammings allowed:	10
Maximum amount of programming failures:	5 %
Total number of devices:	100000
Serial number start value:	22330000

Email notification settings

SMTP server:	smtp.company.com
SMTP auth username:	admin@company.com
SMTP auth password:	12345
From address:	flasher.secure.production@company.com
Send notifications to user:	MyUser (MyUser@company.com)

Send email notfications for the selected events:

Manufacturer blocked	☐
Project finished	☐

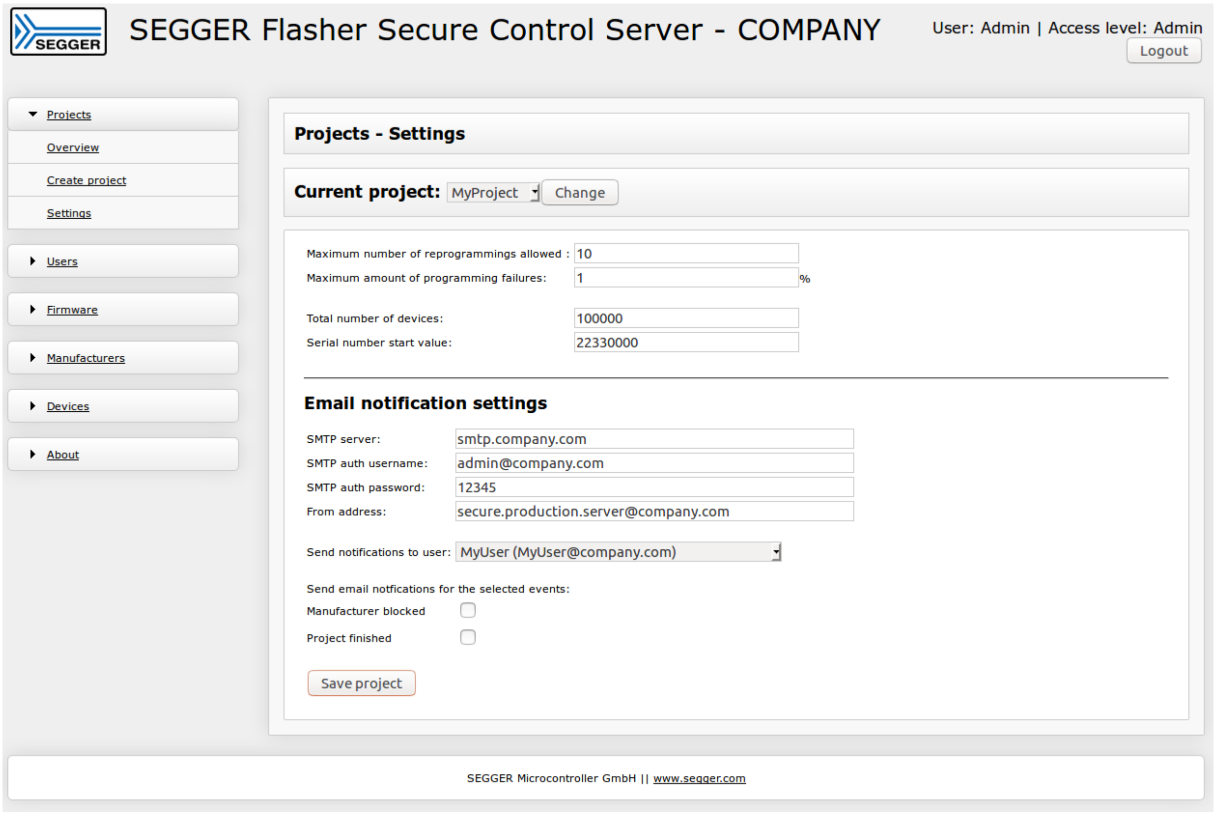
Property	Meaning
Project name	A name for the project. This name is used to identify it in the other menus of the server and on the Flasher Secure unit.
Maximum number of repro-grammings allowed	Defines how often a single device may be reprogrammed.
Maximum amount of pro-gramming failures	Maximum rate of failed device programmings, counted over all programming attempts.
Total number of devices	Maximum number of devices which can be programmed.
Serial number start value	First serial number used for programming.
SMTP server	The email server to which Flasher Secure will connect for sending emails.
SMTP auth username	The user name for the email account.
SMTP auth password	The password for the email account.
Send notifications to user	The email will be sent to the choosen user.
Send email notifications for the selected events	
Manufacturer blocked	Email is send if a manufacturer was blocked event was raised.
Project finished	Email is send if the project is finished event was raised.

3.2.3.2 Project settings

The project settings page provides basic configuration options to the manufacturing process. The first two options control, how to handle attempts to reprogram an already known device and how to deal with programming failures.

The number of reprogramming attempts is limited in order to avoid devices wearing out from too many programming cycles. If a device has been programmed the given times, any further programming attempts will be rejected.

In general, for each device the result of the last programming cycle counts. Over all devices, the failure rate can be limited. If a manufacturer exceeds the configured failure rate, the manufacturer's account is disabled upon clarification.



SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Overview
Create project
Settings
Users
Firmware
Manufacturers
Devices
About
Projects - Settings
Current project: MyProject
Change
Save project
SEGGER Microcontroller GmbH || www.segger.com

Maximum number of reprogrammings allowed :	10
Maximum amount of programming failures:	1 %
Total number of devices:	100000
Serial number start value:	22330000

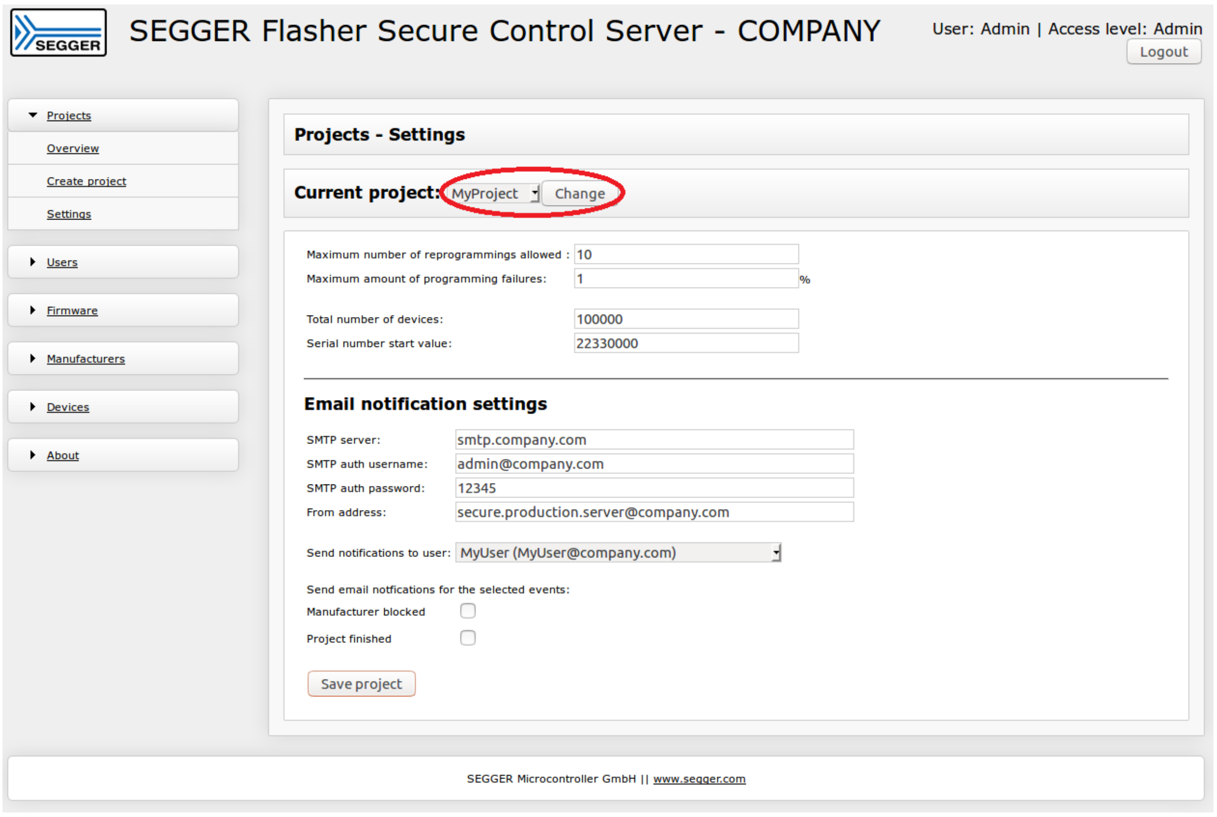
Email notification settings

SMTP server:	smtp.company.com
SMTP auth username:	admin@company.com
SMTP auth password:	12345
From address:	secure.production.server@company.com
Send notifications to user:	MyUser (MyUser@company.com)

Send email notfications for the selected events:

Manufacturer blocked	☐
Project finished	☐

The project can be choosen by selecting the project in the marked list and pressing the change button.



SEGGER Flasher Secure Control Server - COMPANY

User: Admin | Access level: Admin

Projects - Settings

Current project: MyProject

Maximum number of reprogrammings allowed :	10
Maximum amount of programming failures:	1 %
Total number of devices:	100000
Serial number start value:	22330000

Email notification settings

SMTP server:	smtp.company.com
SMTP auth username:	admin@company.com
SMTP auth password:	12345
From address:	secure.production.server@company.com
Send notifications to user:	MyUser (MyUser@company.com)

Send email notfications for the selected events:

Manufacturer blocked	☐
Project finished	☐

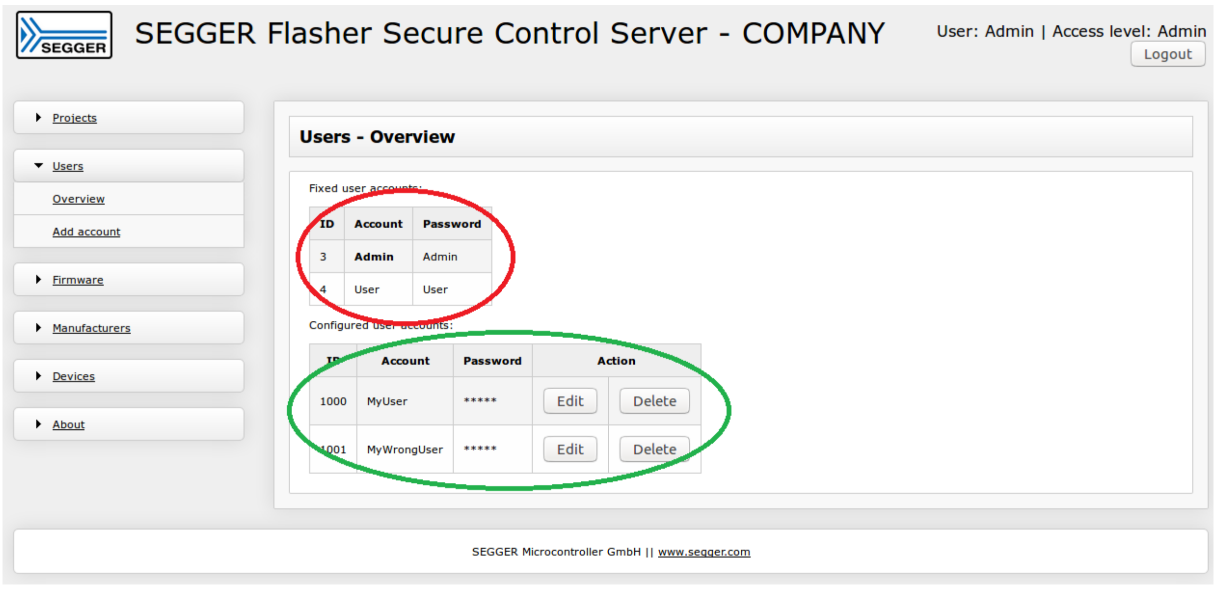
SEGGER Microcontroller GmbH || www.segger.com

3.2.4 User configuration

The current user configuration is listed on the user configuration page. There are two ways to add user accounts to the Flasher Secure Server:

- using the web interface and the Add user login page or
- the configuration file `CFG_USER.TXT` located in the `[installation-directory]/DATA/Company`.

The figure shows a sample user configuration. The red marked user accounts are defined in the configuration file. They cannot be changed or removed within the web interface. The user accounts marked within the green circle can be changed in the web interface.



Users - Overview

Fixed user accounts:

ID	Account	Password
3	Admin	Admin
4	User	User

Configured user accounts:

ID	Account	Password	Action
1000	MyUser	*****	Edit Delete
1001	MyWrongUser	*****	Edit Delete

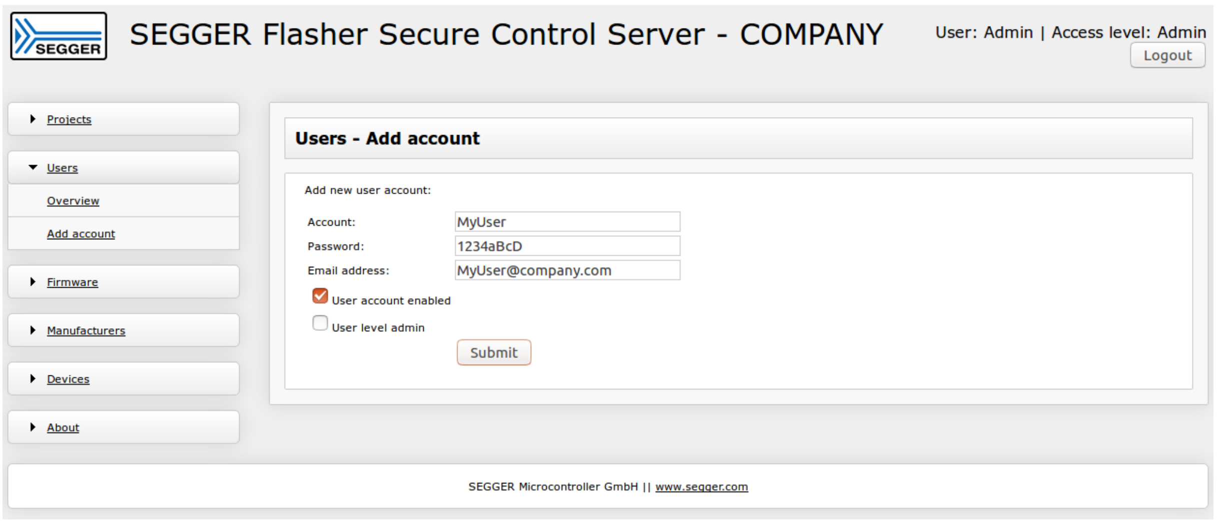
SEGGER Microcontroller GmbH || www.segger.com

The configuration file contains the default users.

There are two access levels: Admin and User. While administrators are allowed to change anything, users only have limited access to configuration settings.

> **Note**
>
> The Flasher Secure Server is delivered with two default accounts. We recommend to removed them after creating your own accounts.

3.2.4.1 Adding a user account


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Overview
Add account
Firmware
Manufacturers
Devices
About
SEGGER Microcontroller GmbH || www.segger.com

Users - Add account

Add new user account:

Account:	MyUser
Password:	1234aBcD
Email address:	MyUser@company.com

[✓] User account enabled

[ ] User level admin

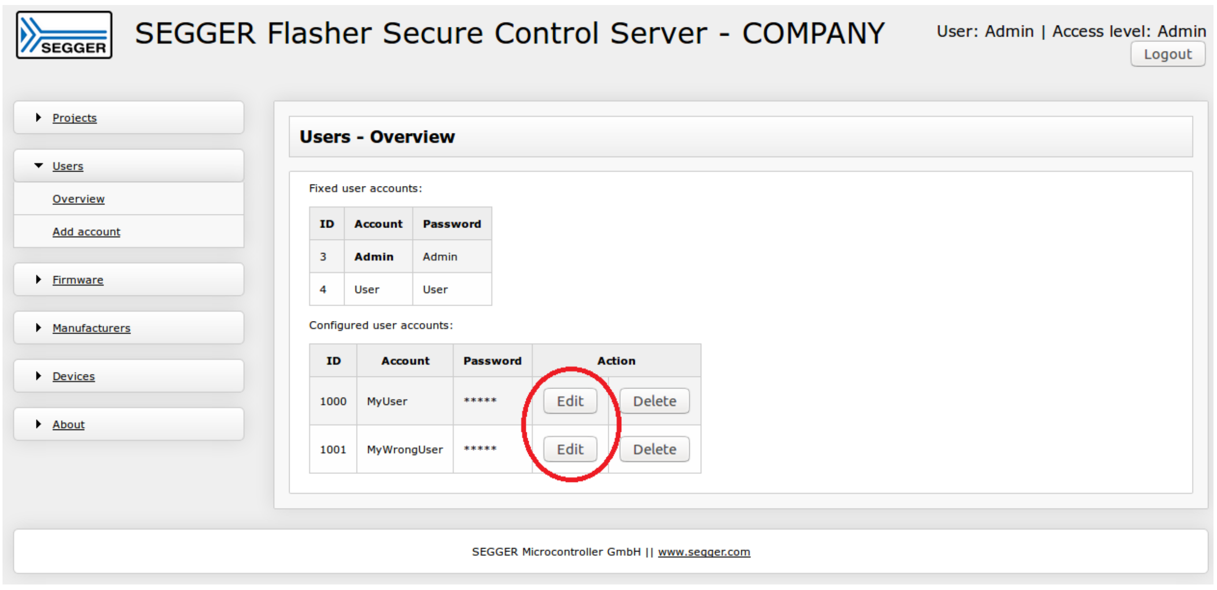
Submit

The following properties can be configured:

Property	Meaning
Account	User name used for the account.
Password	Password used for login.
Email address	Email address used for notifications.
Login enabled	Login is only enabled if checked.
User level Admin	User has administration access level if checked.

3.2.4.2 Changing a user account

The user account can be changed anytime. Select the user account by pressing the edit button, see figure.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Overview
Add account
Firmware
Manufacturers
Devices
About
SEGGER Microcontroller GmbH || www.segger.com

Users - Overview

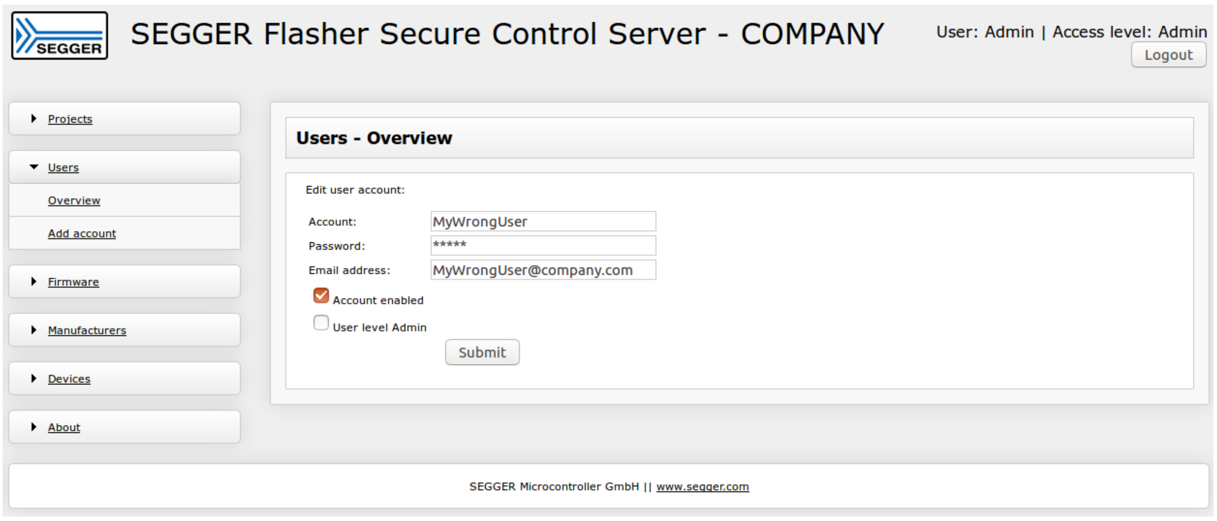
Fixed user accounts:

ID	Account	Password
3	**Admin**	Admin
4	User	User

Configured user accounts:

ID	Account	Password	Action	
1000	MyUser	*****	Edit	Delete
1001	MyWrongUser	*****	Edit	Delete

And on the user setting page you can change every setting for the user.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Overview
Add account
Firmware
Manufacturers
Devices
About
Users - Overview
Submit
SEGGER Microcontroller GmbH || www.segger.com

Edit user account:

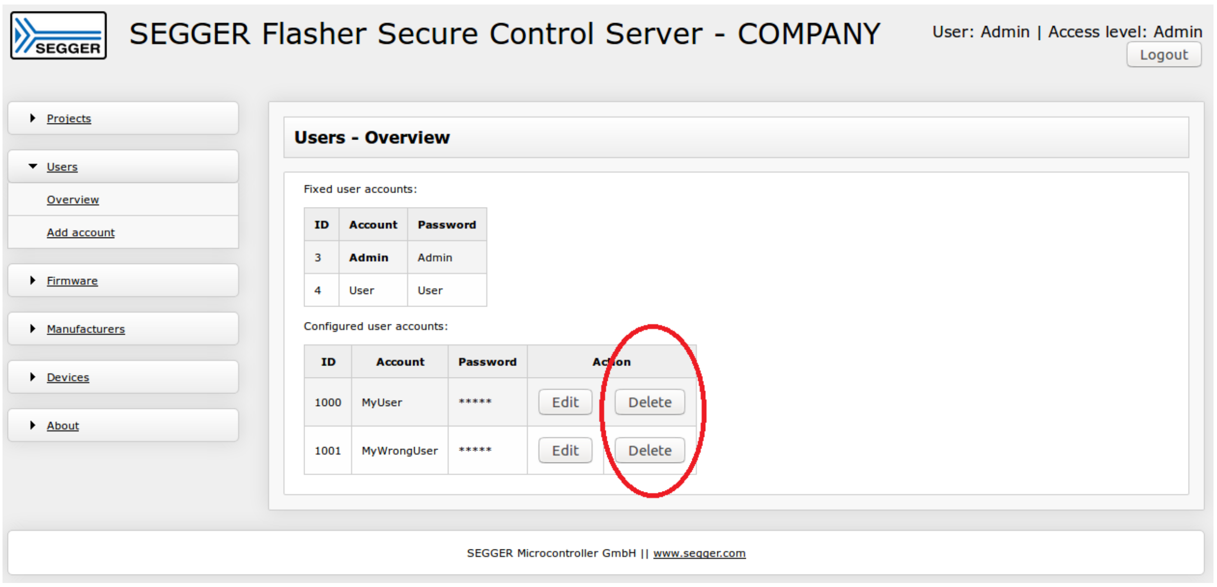
Account:	MyWrongUser
Password:	*****
Email address:	MyWrongUser@company.com

[✓] Account enabled

[ ] User level Admin

3.2.4.3 Deleting a user account

A user account can be deleted anytime. Delete the user account by pressing the delete button, see figure.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Overview
Add account
Firmware
Manufacturers
Devices
About
Users - Overview
SEGGER Microcontroller GmbH || www.segger.com

Fixed user accounts:

ID	Account	Password
3	**Admin**	Admin
4	User	User

Configured user accounts:

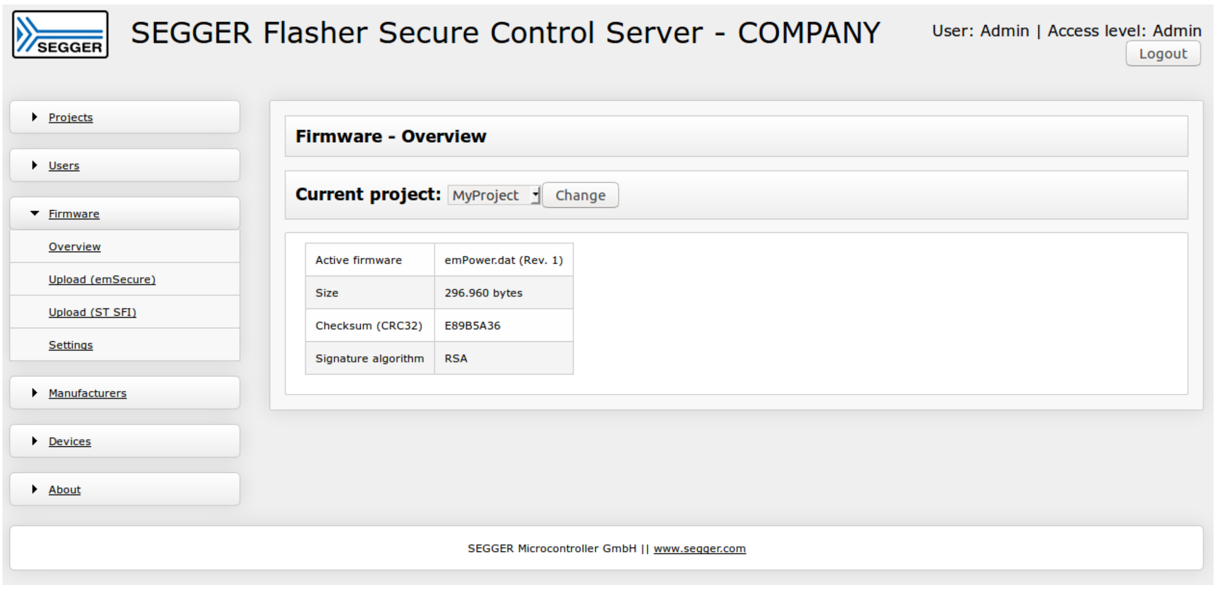
ID	Account	Password	Action	
1000	MyUser	*****	Edit	Delete
1001	MyWrongUser	*****	Edit	Delete

If you only want to deactivate a user account temporarily, you can set the **Login enabled** property to disabled on the user configuration page.

3.2.5 Firmware

3.2.5.1 Firmware Overview

The overview page shows the currently used firmware image properties for the selected project.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Overview
Upload (emSecure)
Upload (ST SFI)
Settings
Manufacturers
Devices
About
Firmware - Overview
Current project: MyProject
Change
SEGGER Microcontroller GmbH || www.segger.com

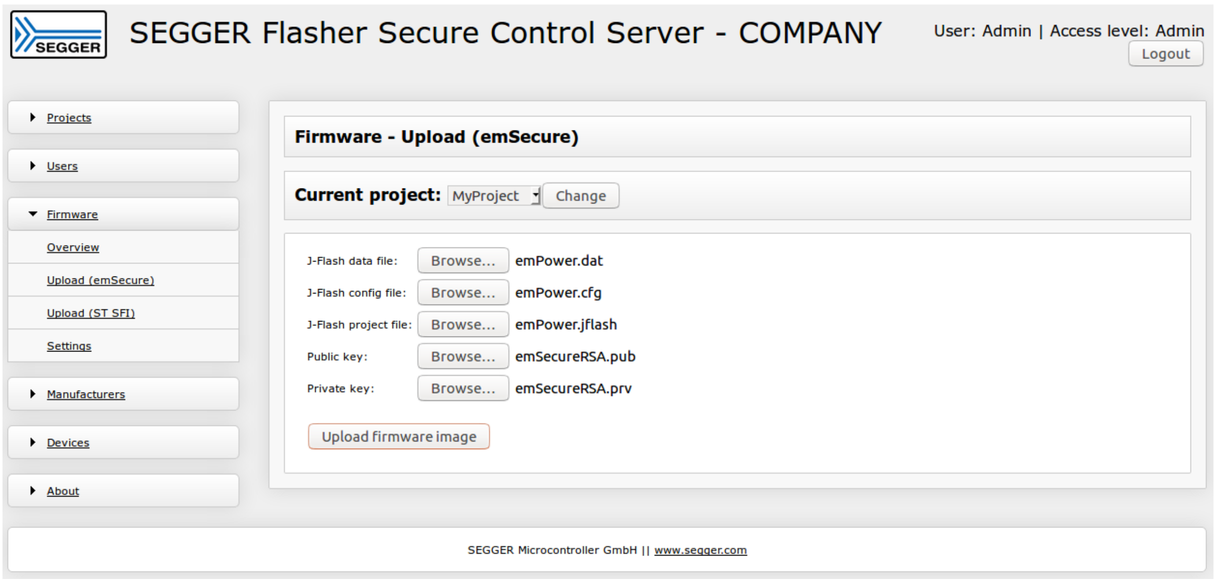
Active firmware	emPower.dat (Rev. 1)
Size	296.960 bytes
Checksum (CRC32)	E89B5A36
Signature algorithm	RSA

You can switch between the projects using the drop down list and the change button.

3.2.5.2 Firmware upload (emSecure)

This page provides the upload interface for the firmware image files when using the emSecure programming method. The server uses both the private and the public key file to generate and verify a signature. The key files are created by a key generator included in the RSA or ECDSA package.

The server automatically detects the encryption standard based on of the key files.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Overview
Upload (emSecure)
Upload (ST SFI)
Settings
Manufacturers
Devices
About
Firmware - Upload (emSecure)
Browse...
Upload firmware image
SEGGER Microcontroller GmbH || www.segger.com

Current project: MyProject Change

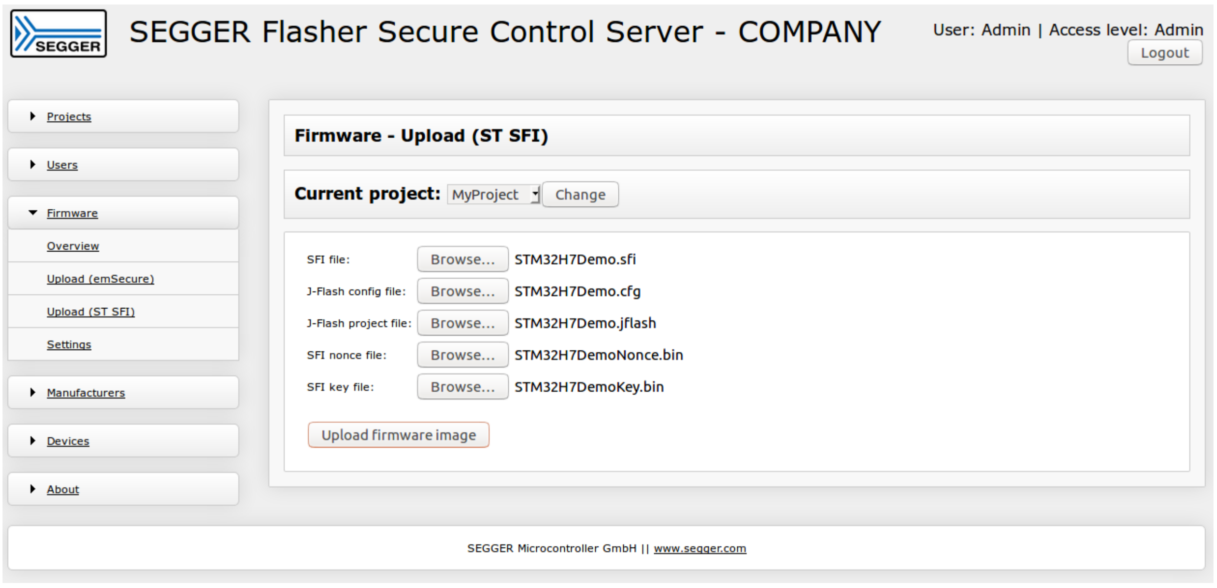
J-Flash data file:	Browse...	emPower.dat
J-Flash config file:	Browse...	emPower.cfg
J-Flash project file:	Browse...	emPower.jflash
Public key:	Browse...	emSecureRSA.pub
Private key:	Browse...	emSecureRSA.prv

Note

You need to prepare the firmware image with the JFlash tool for the Flasher Secure outside the Flasher Secure Server. The latest version of the tool is available https://www.segger.com/downloads/jlink. For the usage of the JFlash tool please read the manual UM08003 available form the same page or this link https://www.segger.com/downloads/jlink/UM08003.

3.2.5.3 Firmware upload (ST SFI)

This page provides the upload interface for the firmware image files when using ST's SFI programming method. Instead of a regular data file, the firmware is encapsulated in an encrypted container file (SFI file). The server additionally needs the "nonce file" and the "key file" in order to create the unique license file required to program the target device.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Overview
Upload (emSecure)
Upload (ST SFI)
Settings
Manufacturers
Devices
About
Firmware - Upload (ST SFI)
Change
Browse...
Upload firmware image
SEGGER Microcontroller GmbH || www.segger.com

Current project: MyProject

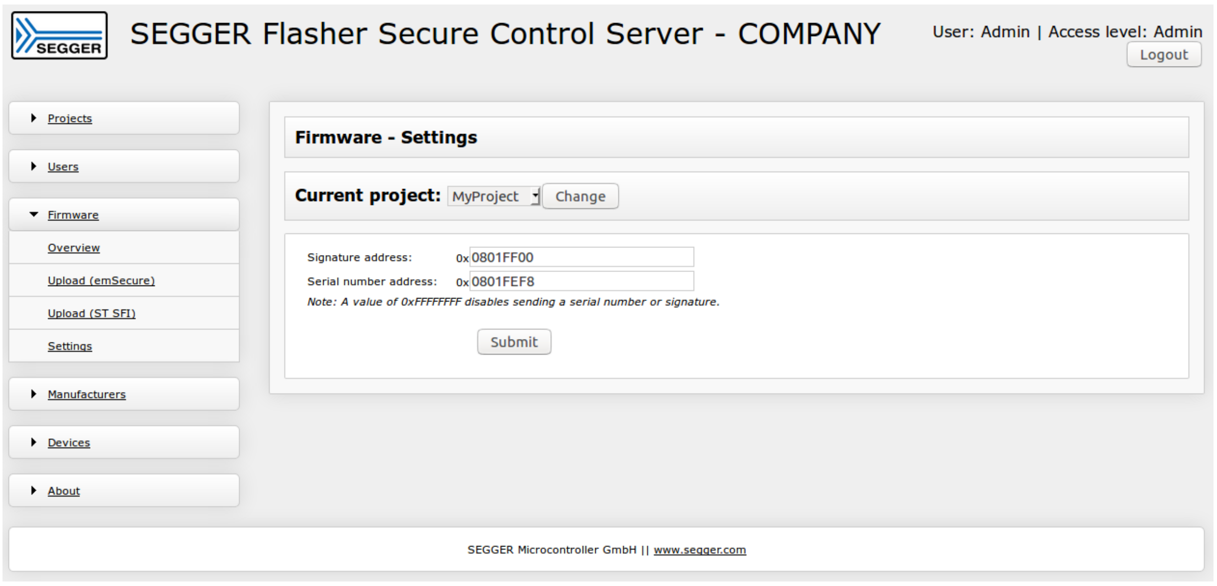
SFI file:	STM32H7Demo.sfi
J-Flash config file:	STM32H7Demo.cfg
J-Flash project file:	STM32H7Demo.jflash
SFI nonce file:	STM32H7DemoNonce.bin
SFI key file:	STM32H7DemoKey.bin

Note

You need to prepare the firmware image with a tool available from ST. Please contact ST for more details and the toolkit for SFI flash programming.

3.2.5.4 Firmware settings

The Flaser Secure needs to know at which address in the target memory the serial number and the signature should be programmed. These addresses needs to be specified on this page. If the address `0xFFFFFFFF` is specified, the corresponding data is not programmed into the device.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Overview
Upload (emSecure)
Upload (ST SFI)
Settings
Manufacturers
Devices
About
Firmware - Settings
Current project: MyProject
Change
Submit
SEGGER Microcontroller GmbH || www.segger.com

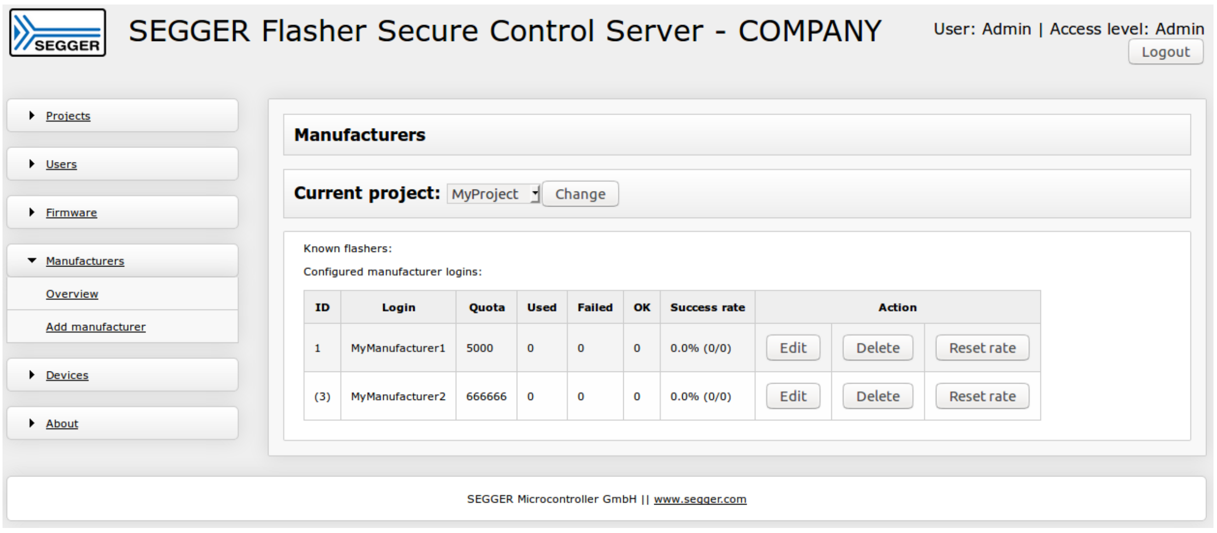
Signature address: 0x 0801FF00
Serial number address: 0x 0801FEF8
Note: A value of 0xFFFFFFFF disables sending a serial number or signature.

Note

The firmware image file needs to contain some dummy data at the target memory range. Otherwise the data cannot be programmed correctly.

3.2.6 Manufacturers

This topic allows you to manage your manufacturers. The overview shows all manufacturers for the selected project and the produces device, the OK and failure rates.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Manufacturers
Overview
Add manufacturer
Devices
About
Manufacturers
Current project: MyProject
Change
SEGGER Microcontroller GmbH || www.segger.com

Known flashers:

Configured manufacturer logins:

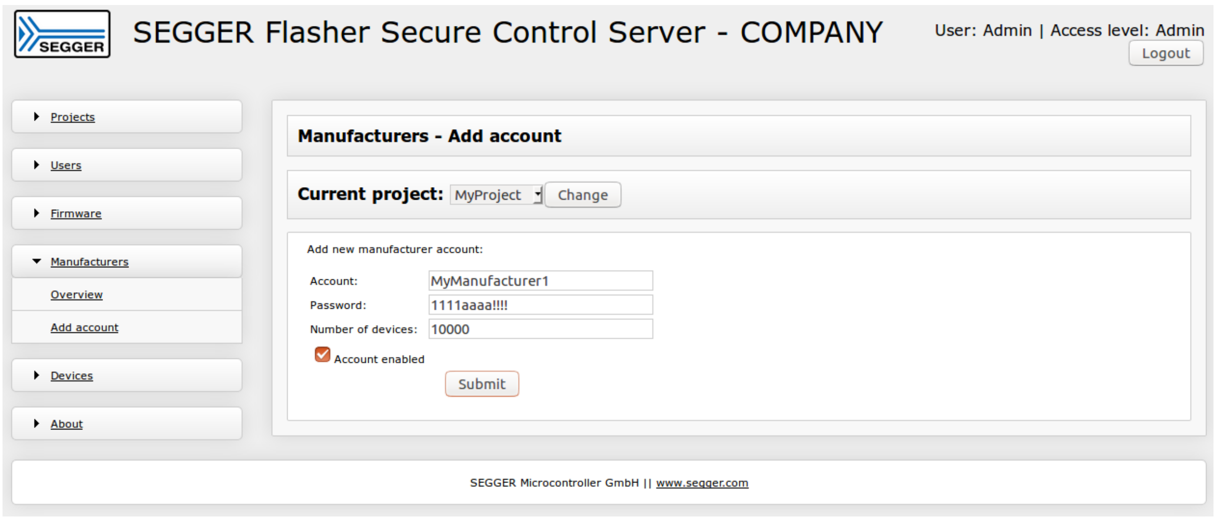
ID	Login	Quota	Used	Failed	OK	Success rate	Action		
1	MyManufacturer1	5000	0	0	0	0.0% (0/0)	Edit	Delete	Reset rate
(3)	MyManufacturer2	666666	0	0	0	0.0% (0/0)	Edit	Delete	Reset rate

Note

Manufacturer’s accounts are not global but linked to one project, so a manufacturer has a separate login to each project. The Flasher Secure Server also keeps track of the used Flasher Secure Devices. Any known Flasher Secure Device, together with manufacturer and date, is shown in the project, where it was used.

3.2.6.1 Add manufacturer account

This page provides the possibility to add a manufacturer for a project. Choose the project by using the drop down list and change button.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Manufacturers
Overview
Add account
Devices
About
Manufacturers - Add account
Current project: MyProject
Change
Submit
SEGGER Microcontroller GmbH || www.segger.com

Add new manufacturer account:

Account: MyManufacturer1
Password: 1111aaaa!!!!
Number of devices: 10000
✓ Account enabled

The following properties can be configured:

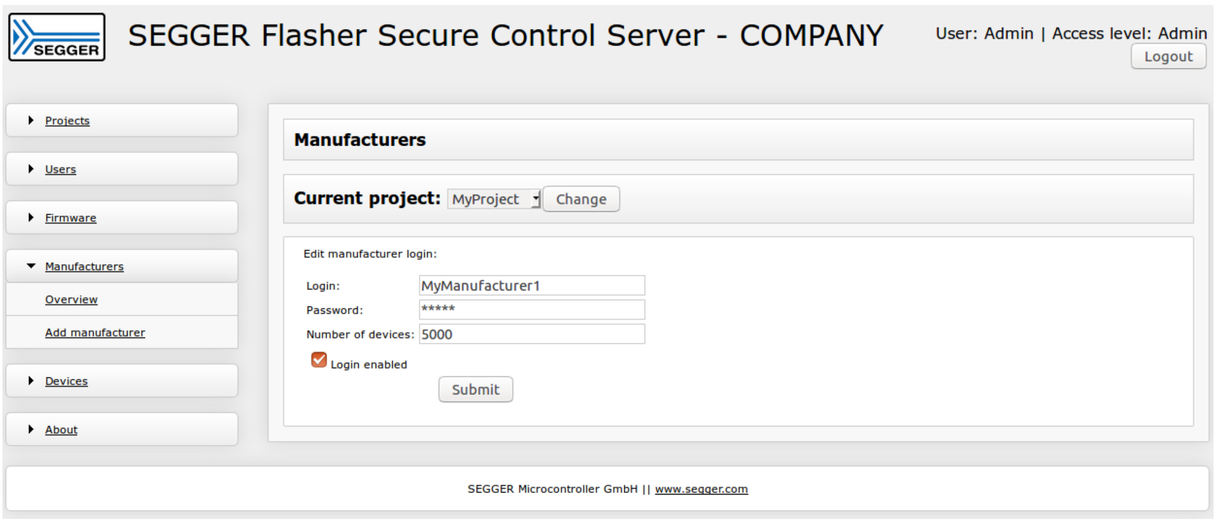
Property	Meaning
Account	User name used for the account or Flasher Secure authentication.
Password	Password used for login or Flasher Secure authentication.
Number of devices	Manufacturers quota on devices.
Login enabled	Login is only enabled if checked.

Note

Manufacturer’s accounts are not global but linked to one project, so a manufacturer has a separate login to each project. The Flasher Secure Server also keeps track of the used Flasher Secure Devices. Any known Flasher Secure Device, together with manufacturer and date, is shown in the project, where it was used.

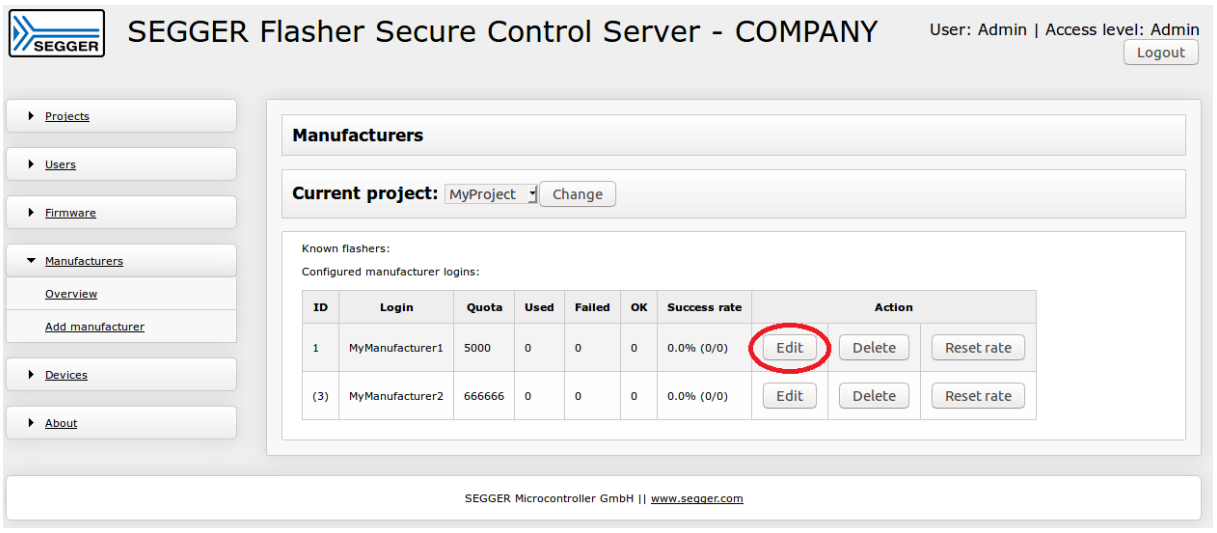
3.2.6.2 Change a manufacturer account

This page provides the possibility to change a manufacturer login. Choose the edit button of the manufacturer you want to alter.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Manufacturers
Overview
Add manufacturer
Devices
About
Manufacturers
Current project: MyProject
Change
Submit
SEGGER Microcontroller GmbH || www.segger.com

Edit manufacturer login:

Login: MyManufacturer1
Password: *****
Number of devices: 5000
✓ Login enabled


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Manufacturers
Overview
Add manufacturer
Devices
About
Manufacturers
Current project: MyProject
Change
SEGGER Microcontroller GmbH || www.segger.com

Known flashers:

Configured manufacturer logins:

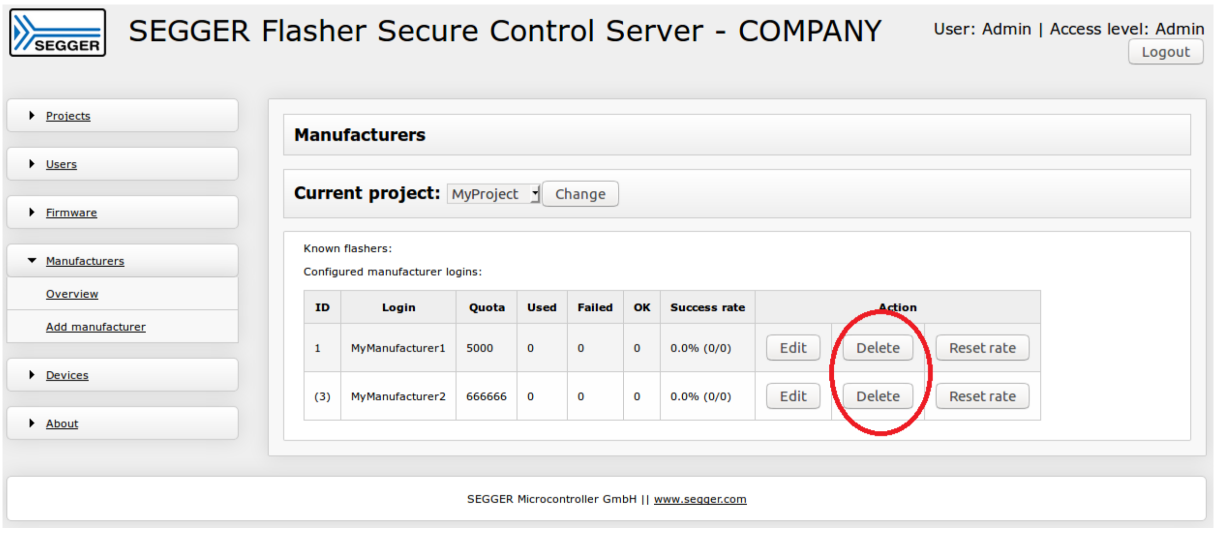
ID	Login	Quota	Used	Failed	OK	Success rate	Action		
1	MyManufacturer1	5000	0	0	0	0.0% (0/0)	Edit	Delete	Reset rate
(3)	MyManufacturer2	666666	0	0	0	0.0% (0/0)	Edit	Delete	Reset rate

The following properties can be configured:

Property	Meaning
Account	User name used for the account or Flasher Secure authentication.
Password	Password used for login or Flasher Secure authentication.
Number of devices	Manufacturers quota on devices.
Login enabled	Login is only enabled if checked.

3.2.6.3 Delete a manufacturer account

The delete button removes the manufacturer login. You will also lose the statistic of this manufacturer. So it might be more suitable for you only to deactivate the login for this manufacturer in the manufacturer settings (see *Change a manufacturer account* on page 35).



Manufacturers

Current project: MyProject Change

Known flashers:

Configured manufacturer logins:

ID	Login	Quota	Used	Failed	OK	Success rate	Action		
1	MyManufacturer1	5000	0	0	0	0.0% (0/0)	Edit	Delete	Reset rate
(3)	MyManufacturer2	666666	0	0	0	0.0% (0/0)	Edit	Delete	Reset rate

SEGGER Microcontroller GmbH || www.segger.com

3.2.6.4 Reset a manufacturer's success rate counter

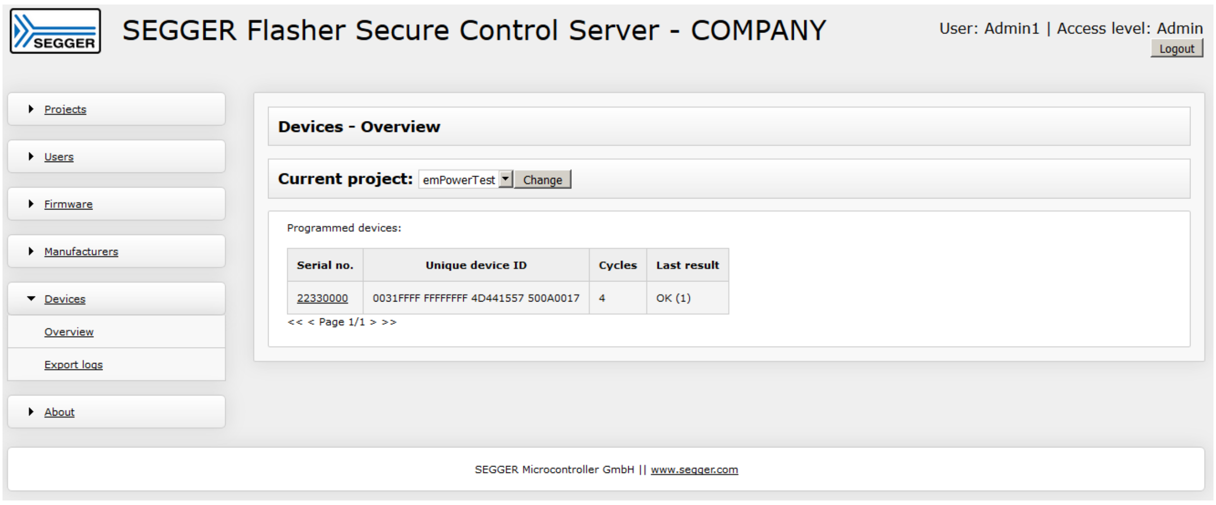
The “Reset rate” button resets the internal counter for measuring the success rate of the manufacturer. Manufacturer’s accounts are disabled automatically, if the manufacturer’s yield drops below a specific threshold value set in the project configuration. To re-enable the manufacturer’s account, the rate has to be reset using the “Reset rate” button.

3.2.7 Devices

3.2.7.1 Devices overview

The devices overview lists all manufactured devices. The list contains for each device the following parameters:

- running production number,
- serial number,
- unique device ID,
- programming cycles and
- last programming result.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin1 | Access level: Admin
Logout
Projects
Users
Firmware
Manufacturers
Devices
Overview
Export logs
About
Change
SEGGER Microcontroller GmbH || www.segger.com

Devices - Overview

Current project: emPowerTest

Programmed devices:

Serial no.	Unique device ID	Cycles	Last result
22330000	0031FFFF FFFFFFFF 4D441557 500A0017	4	OK (1)

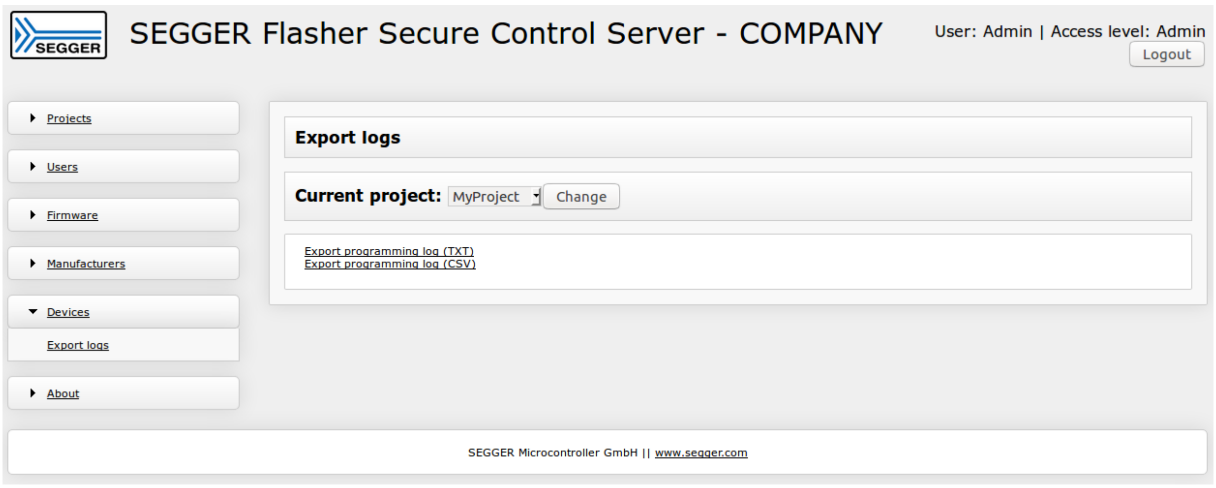
<< < Page 1/1 > >>

The project can be changed using the drop down list and the change button.

3.2.7.2 Export Device log

On this page you can export the device log. Supported formats for the report are:

- csv file or
- plain text file.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Manufacturers
Devices
Export logs
About
Change
SEGGER Microcontroller GmbH || www.segger.com

Export logs

Current project: MyProject

Export programming log (TXT)
Export programming log (CSV)

Plain text format example:

```
2017-03-20 10:23:26 - Programming device 22330000 with unique ID 001B0047
 34345109 35353835 00000000 using Flasher 5: OK (1)
```

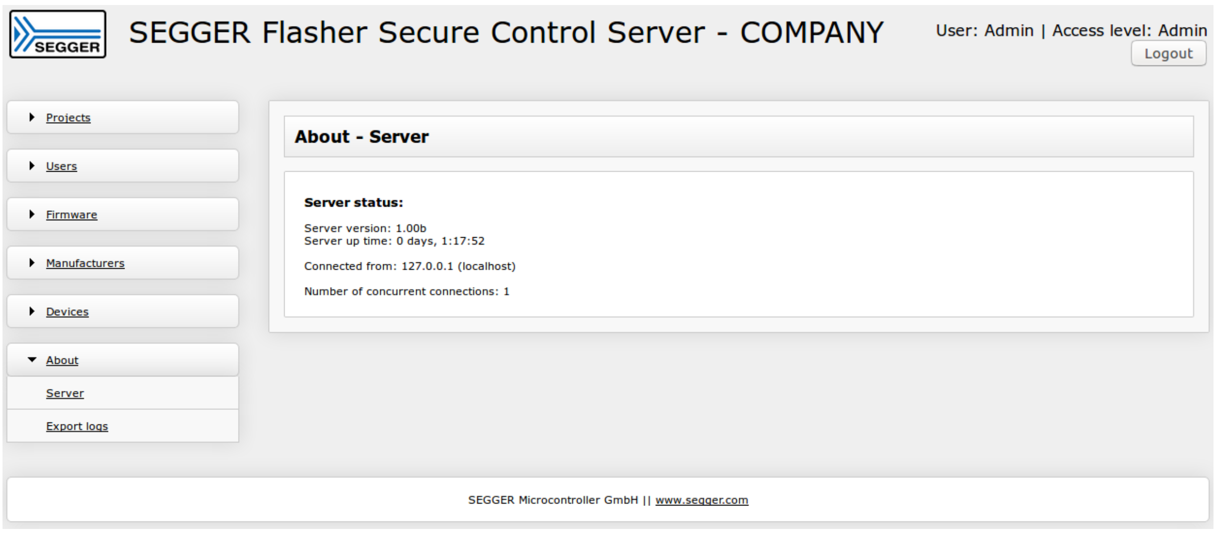
```
2017-03-20 11:13:30 - Programming device 22330001 with unique ID 0039004E
 34345109 35353835 00000000 using Flasher 11: OK (1)
```

CSV format example:

```
"Date","Device S/N","Device UID","Flasher S/N","Result"
"2017-03-20 10:23:26","22330000","47001B0009513434353835350000000","5","1"
"2017-03-20 11:13:30","22330001","4E0039000951343435383535000000000","11","1"
```

3.2.8 Server status

The server status page gives an overview about the current server status.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Manufacturers
Devices
About
Server
Export logs
About - Server
SEGGER Microcontroller GmbH || www.segger.com

Server status:

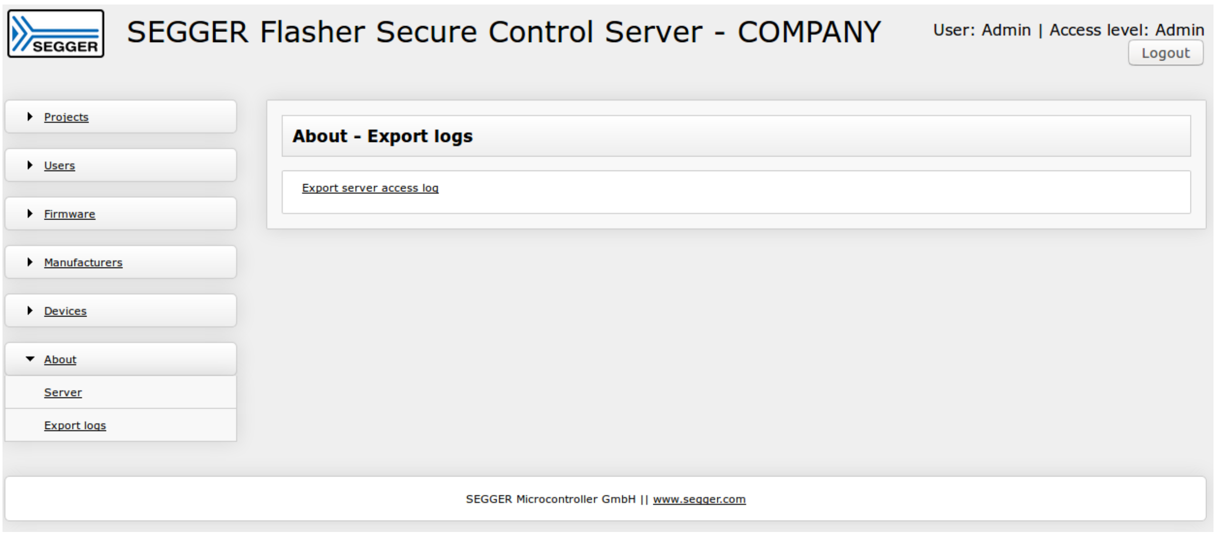
Server version: 1.00b
Server up time: 0 days, 1:17:52

Connected from: 127.0.0.1 (localhost)

Number of concurrent connections: 1

3.2.8.1 Server Access Log

The server's access log, which is also stored on the server as a text file, can be downloaded from the "Export logs" page.


SEGGER
SEGGER Flasher Secure Control Server - COMPANY
User: Admin | Access level: Admin
Logout
Projects
Users
Firmware
Manufacturers
Devices
About
Server
Export logs
About - Export logs
Export server access log
SEGGER Microcontroller GmbH || www.segger.com

The log file is in the NCSA Common log format. The describtion of the format can be found here: http://publib.boulder.ibm.com/tividd/td/ITWSA/ITWSA_info45/en_US/HTML/guide/c-logs.html#common

Access log sample:

```
123.45.67.89 - - [2017-01-01 00:00:00] "GET /"
123.45.67.89 - - [2017-01-01 00:00:00] "GET /style.css"
123.45.67.89 - - [2017-01-01 00:00:00] "GET /img/SeggerLogo_200x.png"
123.45.67.89 - - [2017-01-01 00:00:00] "GET /favicon.ico"
123.45.67.89 - - [2017-01-01 00:00:05] "POST /index.htm"
```

Chapter 4

Flasher Secure unit

The Flasher Secure unit is based on the SEGGER Flasher unit, so it generally supports the same devices as the Flashers do. Devices, which do not provide a UID or a vendor specific secure programming solution, are not supported.

4.1 General

The Flasher Secure unit is the programming unit, which is connected to the target device.

Note

The Flasher Secure needs an internet connection or network connection. It needs to be able to reach the Flasher Secure Server, which provides the signature generation service.

4.2 IP configuration

IP configuration is done via DHCP by default. There are different configuration options for setting up a manual configuration:

- J-Link Configurator (refer to *UM08001: J-Link / J-Trace User Guide*, chapter *4 Setup*, section *4.5 J-Link Configurator*), short sum up see here *IP configuraton with J-Link Configurator* on page 42
- J-Link Commander (refer to *UM08001: J-Link / J-Trace User Guide*, chapter *3 J-Link software and documentation package*, section *3.2 J-Link Commander*)
- Flasher Secure web interface (*Web interface* on page 21).

Note

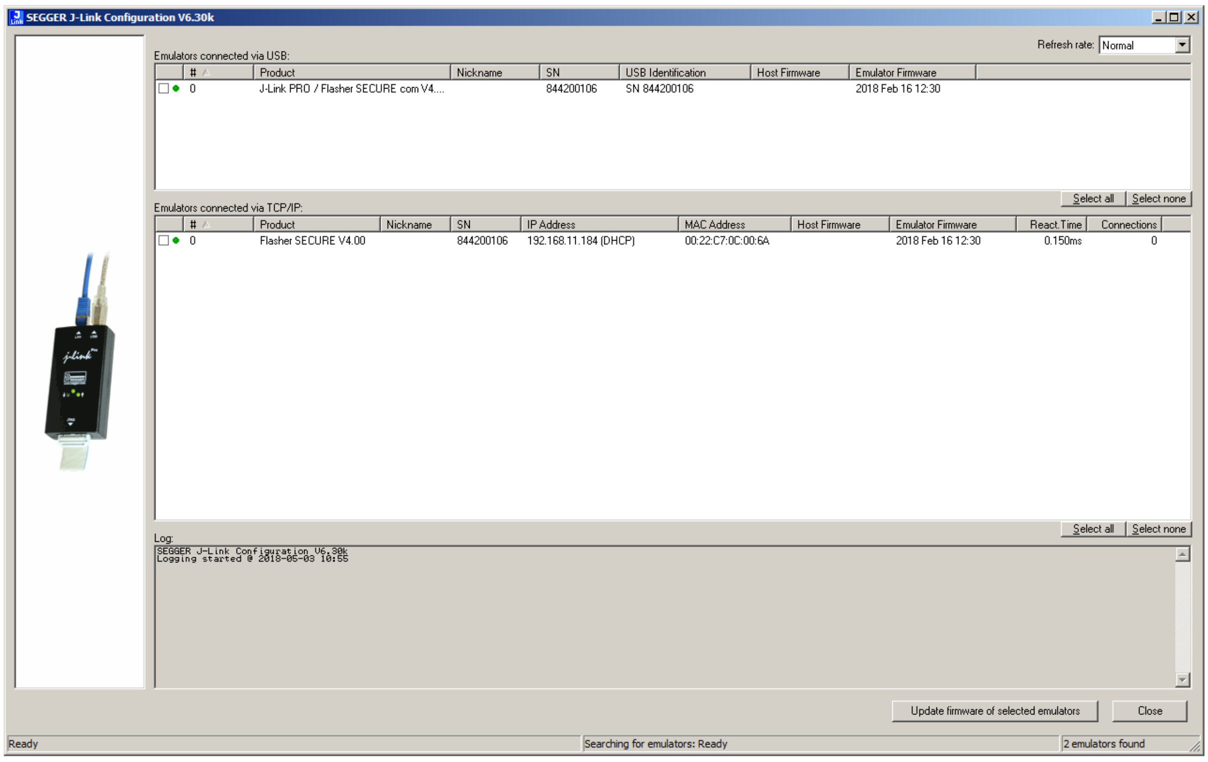
The Flasher Secure unit supports IPv4 only.

4.2.1 Manual IP configuration

4.2.1.1 IP configuraton with J-Link Configurator

Start the J-Link Configurator and wait until your Flasher Secure was found.

Choose your Flasher Secure in the list, e.g. in the IP connected device list and open the context menu with a right click on the device entry.


SEGGER J-Link Configuration V6.30k
Refresh rate: Normal
Select all
Select none
Update firmware of selected emulators
Close

Emulators connected via USB:

	#	Product	Nickname	SN	USB Identification	Host Firmware	Emulator Firmware
☐ ●	0	J-Link PRO / Flasher SECURE com V4....		844200106	SN 844200106		2018 Feb 16 12:30

Emulators connected via TCP/IP:

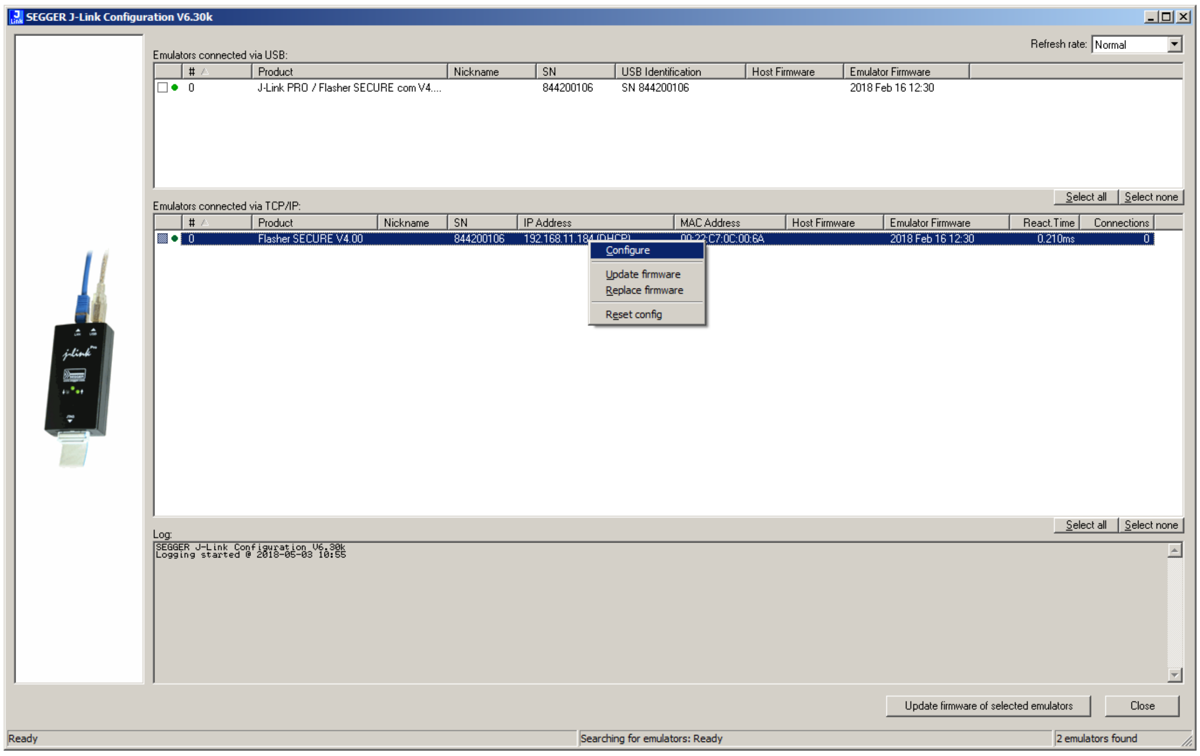
	#	Product	Nickname	SN	IP Address	MAC Address	Host Firmware	Emulator Firmware	React.Time	Connections
☐ ●	0	Flasher SECURE V4.00		844200106	192.168.11.184 (DHCP)	00:22:C7:0C:00:6A		2018 Feb 16 12:30	0.150ms	0

Log:

```
SEGGER J-Link Configuration V6.30k
Logging started @ 2018-05-03 10:55
```

Ready | Searching for emulators: Ready | 2 emulators found

Choose the *Configure* entry in the context menu.



Emulators connected via USB:

	#	Product	Nickname	SN	USB Identification	Host Firmware	Emulator Firmware
☐ ●	0	J-Link PRO / Flasher SECURE com V4....		844200106	SN 844200106		2018 Feb 16 12:30

Emulators connected via TCP/IP:

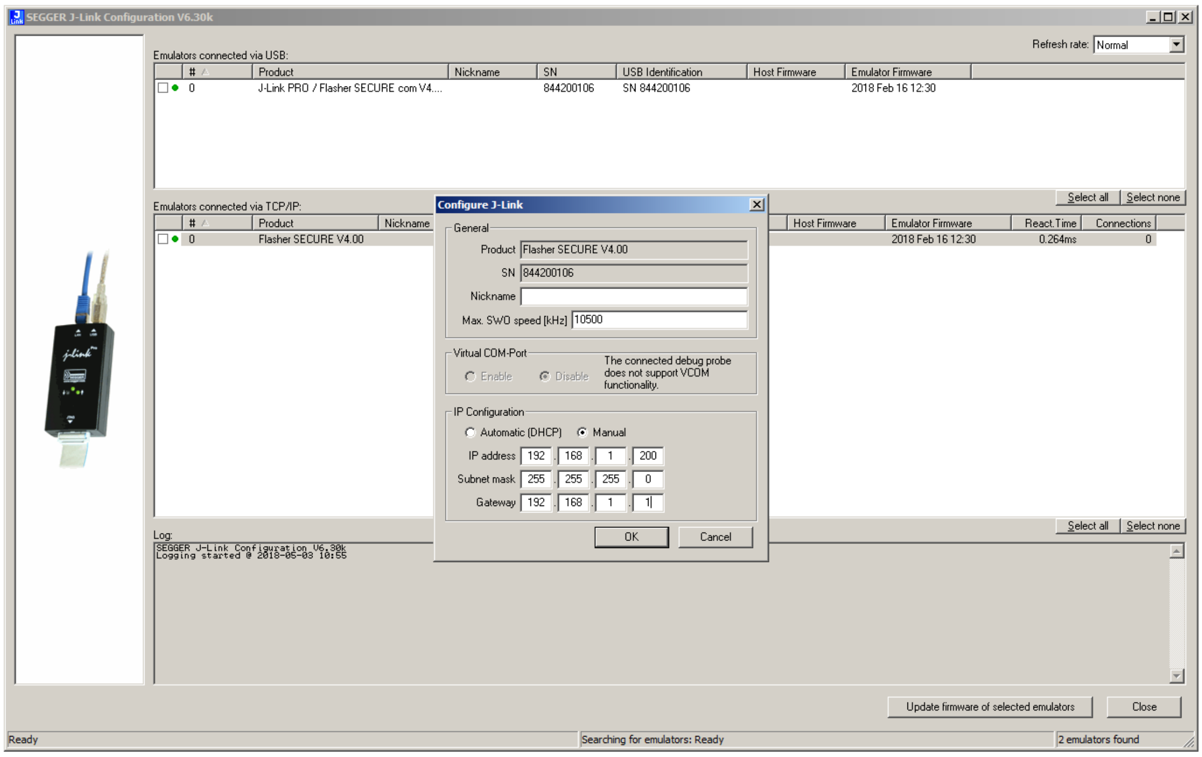
	#	Product	Nickname	SN	IP Address	MAC Address	Host Firmware	Emulator Firmware	React.Time	Connections
▨ ●	0	Flasher SECURE V4.00		844200106	192.168.11.184 (DHCP)	00:22:C7:0C:00:6A		2018 Feb 16 12:30	0.210ms	0

Log:

```
SEGGER J-Link Configuration V6.30k
Logging started @ 2018-05-03 10:55
```

Ready | Searching for emulators: Ready | 2 emulators found

Enter the IP configuration in the pop up menu and confirm them using the OK button.


SEGGER J-Link Configuration V6.30k
Refresh rate: Normal
Select all
Select none
Configure J-Link
OK
Cancel
Update firmware of selected emulators
Close

Emulators connected via USB:

	#	Product	Nickname	SN	USB Identification	Host Firmware	Emulator Firmware
☐ ●	0	J-Link PRO / Flasher SECURE com V4....		844200106	SN 844200106		2018 Feb 16 12:30

Emulators connected via TCP/IP:

	#	Product	Nickname	Host Firmware	Emulator Firmware	React.Time	Connections
☐ ●	0	Flasher SECURE V4.00			2018 Feb 16 12:30	0.264ms	0

General

Product: Flasher SECURE V4.00
SN: 844200106
Nickname:
Max. SWO speed [kHz]: 10500

Virtual COM-Port

Enable [✓ Disable] The connected debug probe does not support VCOM functionality.

IP Configuration

Automatic (DHCP) [✓ Manual]

IP address: 192 . 168 . 1 . 200
Subnet mask: 255 . 255 . 255 . 0
Gateway: 192 . 168 . 1 . 1

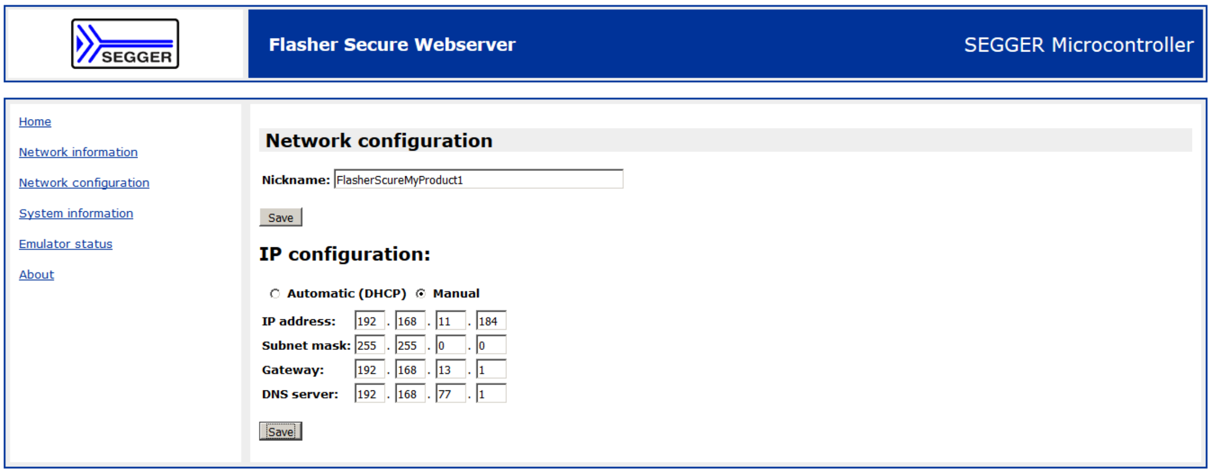
Log:

```
SEGGER J-Link Configuration V6.30k
Logging started @ 2018-05-03 10:55
```

Ready | Searching for emulators: Ready | 2 emulators found

4.2.1.2 IP configuration with integrated web server

Open the browser of your choice. Navigate to your Flasher Secure unit, e.g. *http://192.168.1.200*. Choose the *Network configuration* page in the navigation column on the left side.



Network configuration

Nickname: FlasherScureMyProduct1

IP configuration:

Automatic (DHCP) [✓ Manual]

IP address:	192	168	11	184
Subnet mask:	255	255	0	0
Gateway:	192	168	13	1
DNS server:	192	168	77	1

Enter your IP configuration on the page and confirm it by pressing the save button.

Note

The browse my claim the device is no longer preset, when you changed the IP address. In that case you have to reenter the correct IP address in the navigation bar of the browser.

4.3 Project Configuration

4.3.1 Configuration files

The Flasher Secure unit has an internal and an external storage. The internal storage is about 120MB in size and not accessible from the outside. It is used as cache memory for the firmware and configuration files provided by the server. A 16MB of the internal storage are available for the configuration and log files.

The Flasher Secure needs two configuration files *SECURE.INI* and *SERVERT.CRT*.

A typical configuration file SECURE.INI contains the following information:

```
[SERVER]
Host = "FlahserSecureServer.Company.com"
Port = "3110"

[LOGIN]
Company = "Company"
Project = "MyProject"
Username = "MyManufacturer1"
Password = "1234abcd!!!!"
```

Entry	Meaning
Server section	defines the server settings
Host	the server host name or IP address
Port	the server port
Login section	the login setup
Company	the company name, currently needs to be set to "Company" (reserved for future feature extension)
Project	the project name
Username	the user name for the Flashser Secrue Server login, must be a login of the manufacturers logins.
Password	the password for the login

Note

The SERVER section defines the host and the port of the server to connect to. The host may either be a name or an IP address. If it is a name, a DNS server has to be configured in the IP configuration if necessary. The LOGIN section contains the login data corresponding to a manufacturer account on the server.

The other configuration file SERVER.CRT is the server's TLS certificate. It is used to restrict the Flasher Secure device to only accept connections to servers using the key in the certificate (server identity pinning). Although this file is optional, we recommend to use it. It provides a simple and fool-proof way to avoid man-in-the-middle attacks.

Note

The Flasher Secure supports only RSA certificates for the server validation.

4.3.2 Accessing the configuration files

You have three ways to access the configuration files:

- using the USB MSD mode,
- using the build-in FTP server and

- using the ASCII command interface over the RS232 connector.

4.3.2.1 Access via the USB MSD mode

The 16MB external storage is available as a disk drive, when the Flasher Secure is running in the “MSD mode” (**M**ass **S**torage **D**evice). The Flasher Secure enters this mode, if the button is pressed, you power up the device by connection the USB cable and keep the button pressed for at least 2 seconds.

4.3.2.2 Access via FTP

The Flahser Secure has a build-in FTP server. So you can connect to it using a FTP client of your choice. The FTP server comes with 2 logins.

Login	Password	Privileges
admin	1234	can read, write, upload, download, delete files
anonymous	*none*	can only read or download files

Note

These logins cannot be changed.

4.3.2.3 Access via ASCII command protocol

The files can also be written via the ASCII command protocol. Refer to chapter *Commands and replies* on page 58 and the included file commands.

4.4 Programming devices

The Flasher Secure is typically used to program the target devices in a production setup. Therefore it will erase, flash and secure the target devices in on programming cycle once the setup is configured.

The Flasher Secure can be controlled by four ways:

- using the push button on the device,
- using the 3 wire handshake interface integrated in the RS232 connector or
- using the ASCII command protocol via RS323 or Telnet. For all Options the details are described in chapter *Remote control* on page 54.

A typically programming cycle may look like this for one target device:

Input

```
#AUTO
```

Flasher Secure Response

```
#ACK
#STATUS:INITIALIZING
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#STATUS:PROGRAMMING
#STATUS:VERIFYING
#OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)
```

4.5 LED status indicators

Progress and result of an operation is indicated by Flasher Secure’s LEDs. The behavior is different for J-Link and stand-alone mode. For a definition of the different modes, please refer to *UM08001: J-Link / J-Trace User Guide*, chapter Operating modes.

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

#	Status of LED	Meaning
0	GREEN constant	Flasher Secure waits for a start trigger to perform an operation in stand-alone mode.
1	GREEN slow blinking	Flashing operation in progress: • Erasing (slow blinking on/off time: 80 ms => 6.25 Hz) • Programming (slow blinking on/off time: 300ms => ~1.67 Hz) • Verifying (slow blinking, on/off time: 100ms => 5 Hz)
2	GREEN: constant RED: off or constant	GREEN constant, RED off: Operation successful. GREEN constant, RED constant: Operation failed Goes back to state #0 automatically, but in case of operation failed, RED remains on until state #1 is entered the next time.

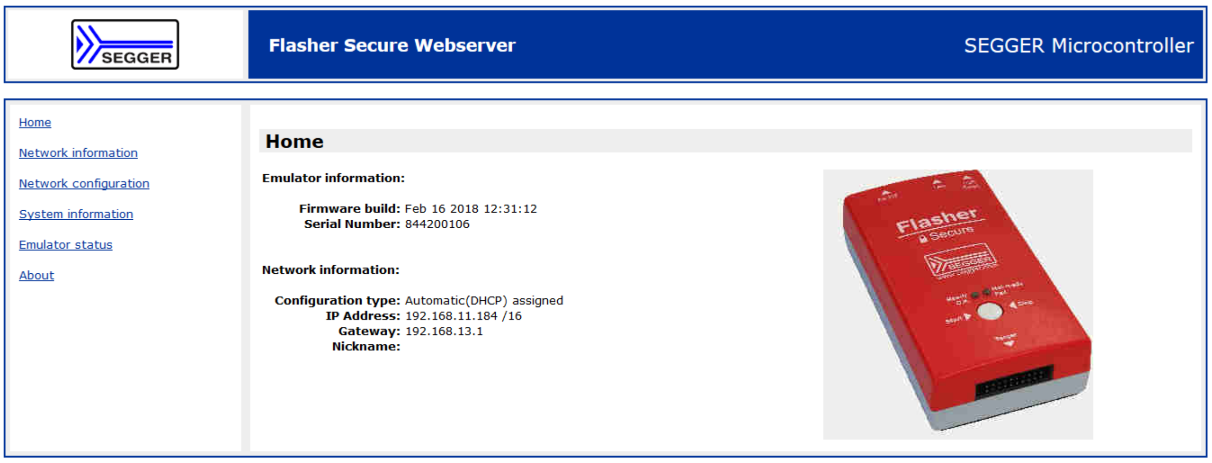
4.6 Web interface

The Flasher Secure has a build-in web interface. This interface shows information of the current status and IP configuration of the unit.

4.6.1 Flasher Secure main page

The Flasher Secure main page shows the following information:

- firmware version,
- serial number,
- network configuration and
- nickname.



Home

Emulator information:

Firmware build: Feb 16 2018 12:31:12
Serial Number: 844200106

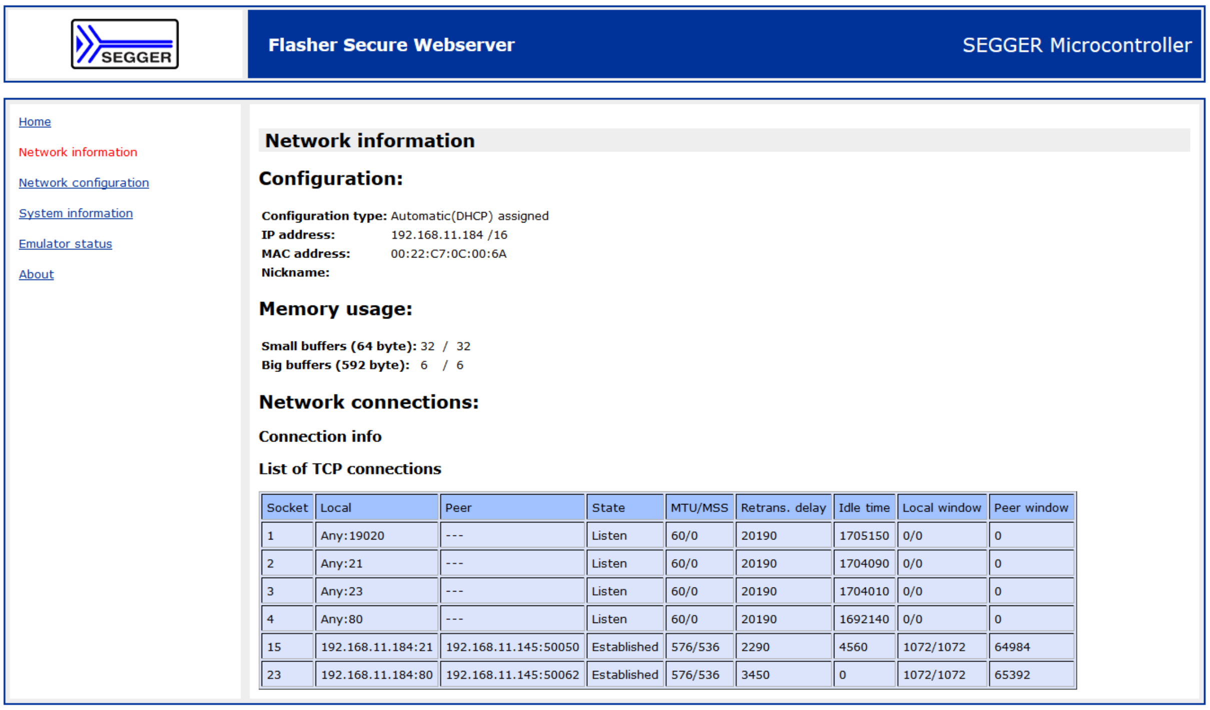
Network information:

Configuration type: Automatic(DHCP) assigned
IP Address: 192.168.11.184 /16
Gateway: 192.168.13.1
Nickname:

4.6.2 Flasher Secure network information page

The Flasher Secure network information page shows the following information:

- IP configuration,
- MAC address,
- nickname,
- memory buffer statistic and
- current network connections.



Network information

Configuration:

Configuration type: Automatic(DHCP) assigned
IP address: 192.168.11.184 /16
MAC address: 00:22:C7:0C:00:6A
Nickname:

Memory usage:

Small buffers (64 byte): 32 / 32
Big buffers (592 byte): 6 / 6

Network connections:

Connection info

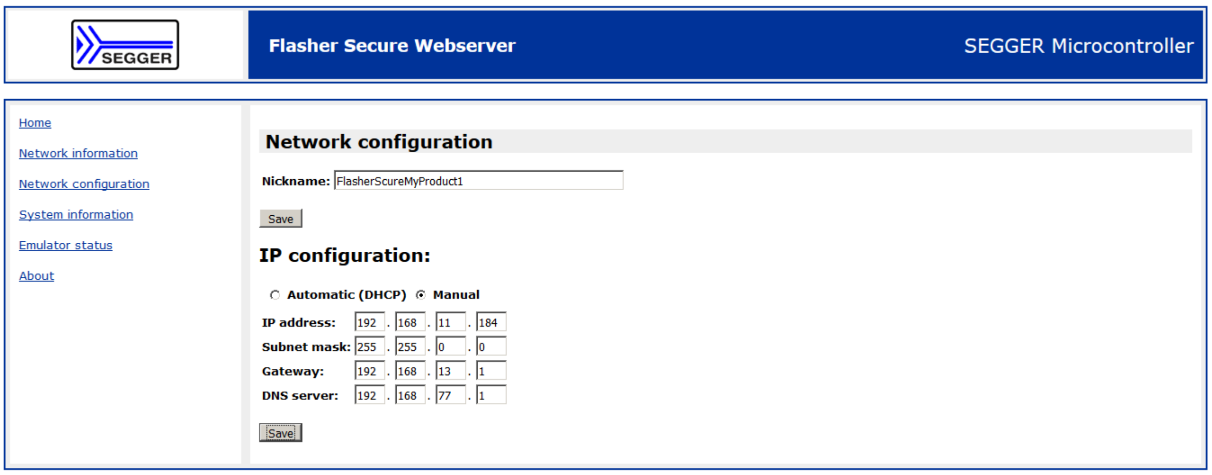
List of TCP connections

Socket	Local	Peer	State	MTU/MSS	Retrans. delay	Idle time	Local window	Peer window
1	Any:19020	---	Listen	60/0	20190	1705150	0/0	0
2	Any:21	---	Listen	60/0	20190	1704090	0/0	0
3	Any:23	---	Listen	60/0	20190	1704010	0/0	0
4	Any:80	---	Listen	60/0	20190	1692140	0/0	0
15	192.168.11.184:21	192.168.11.145:50050	Established	576/536	2290	4560	1072/1072	64984
23	192.168.11.184:80	192.168.11.145:50062	Established	576/536	3450	0	1072/1072	65392

4.6.3 Flasher Secure network configuration

The Flasher Secure network configuration allows the setup of the network settings:

- IP address,
- subnet mask,
- gateway IP address and
- nickname.



Network configuration

Nickname: FlasherScureMyProduct1

IP configuration:

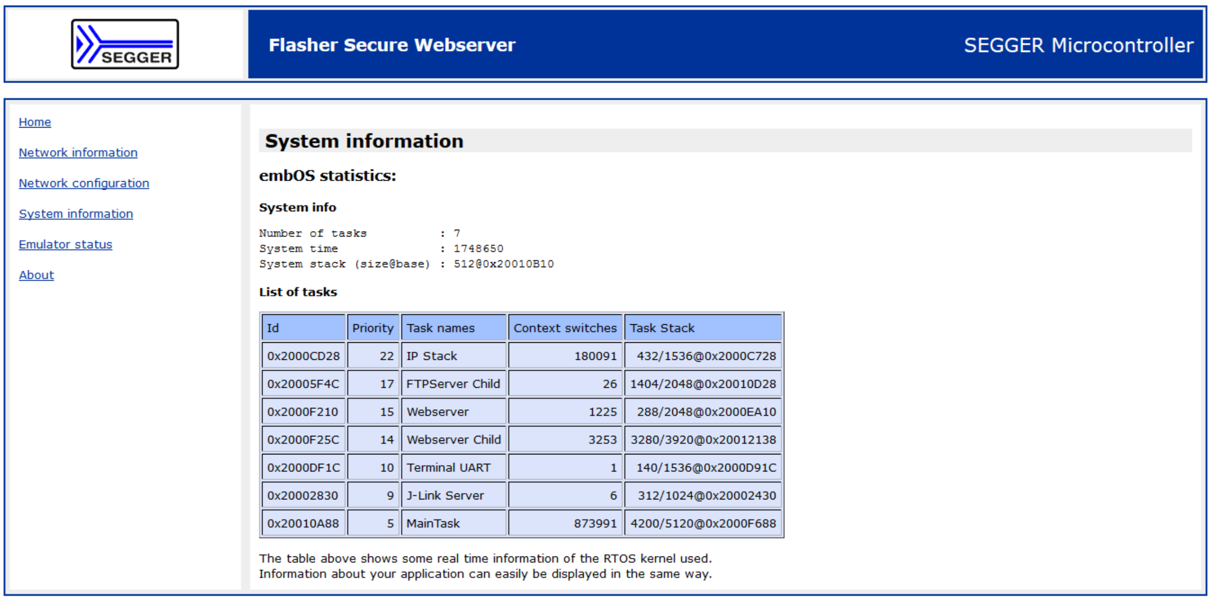
Automatic (DHCP) [✓ Manual]

IP address:	192	168	11	184
Subnet mask:	255	255	0	0
Gateway:	192	168	13	1
DNS server:	192	168	77	1

4.6.4 Flasher Secure system information page

The Flasher Secure system information page lists the following points:

- OS statistic and
- task statistic.



System information

embOS statistics:

System info

```
Number of tasks          : 7
System time              : 1748650
System stack (size@base) : 512@0x20010B10
```

List of tasks

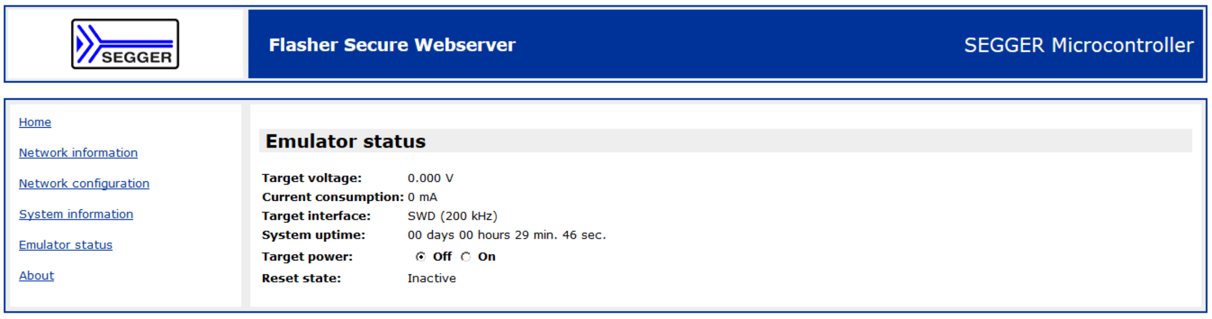
Id	Priority	Task names	Context switches	Task Stack
0x2000CD28	22	IP Stack	180091	432/1536@0x2000C728
0x20005F4C	17	FTPServer Child	26	1404/2048@0x20010D28
0x2000F210	15	Webserver	1225	288/2048@0x2000EA10
0x2000F25C	14	Webserver Child	3253	3280/3920@0x20012138
0x2000DF1C	10	Terminal UART	1	140/1536@0x2000D91C
0x20002830	9	J-Link Server	6	312/1024@0x20002430
0x20010A88	5	MainTask	873991	4200/5120@0x2000F688

The table above shows some real time information of the RTOS kernel used.
Information about your application can easily be displayed in the same way.

4.6.5 Flasher Secure emulator status page

The Flasher Secure emulator status page lists the following points:

- target voltage (VTref),
- target power consumption,
- used target interface and
- target power and reset status.

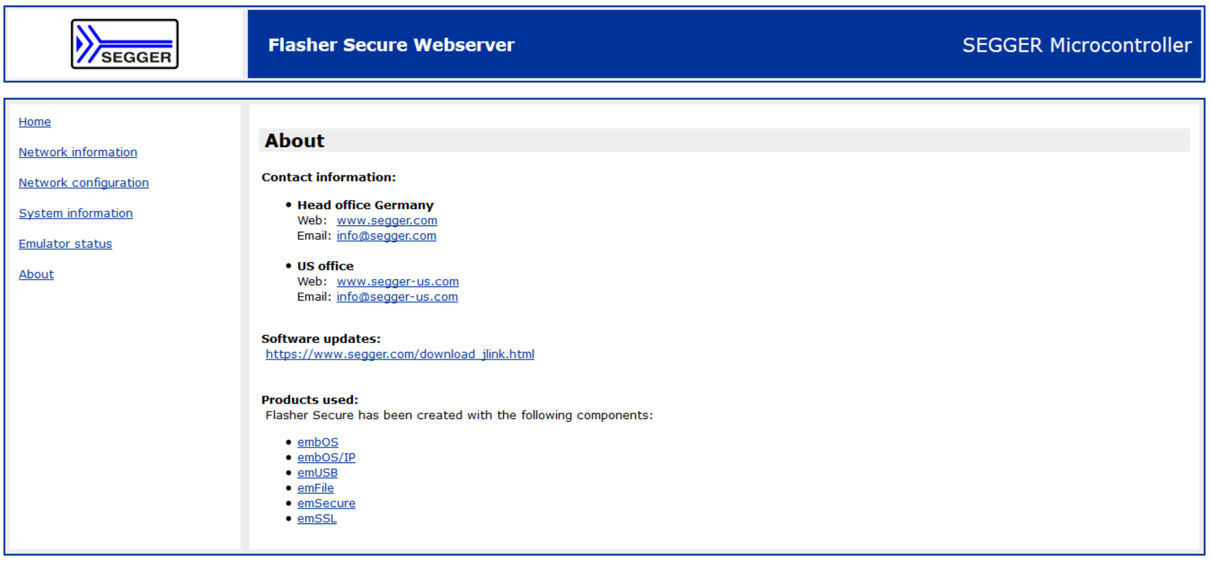


Emulator status

Target voltage:	0.000 V
Current consumption:	0 mA
Target interface:	SWD (200 kHz)
System uptime:	00 days 00 hours 29 min. 46 sec.
Target power:	[✓ Off] On
Reset state:	Inactive

4.6.6 Flasher Secure about page

The Flasher Secure about page lists links to additional information on the SEGGER web presentation.



About

Contact information:

- **Head office Germany**
  Web: www.segger.com
  Email: info@segger.com
- **US office**
  Web: www.segger-us.com
  Email: info@segger-us.com

Software updates:
https://www.segger.com/download_jlink.html

Products used:
Flasher Secure has been created with the following components:

- embOS
- embOS/IP
- emUSB
- emFile
- emSecure
- emSSL

4.7 Additional Information

4.7.1 Patch file support

This feature is currently not supported by Flasher Secure.

4.7.2 J-Link mode

The Flasher Secure can operate in J-Link mode. For further details please have a look into the J-Link Manual (UM08001).

Note

In J-Link mode, the Flasher Secure behaves like a normal J-Link. The Flasher Secure programming process, including the signature generation, is only available in stand-alone mode programming.

Chapter 5

Remote control

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

5.1 Overview

There are 4 ways to control Flasher operation:

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

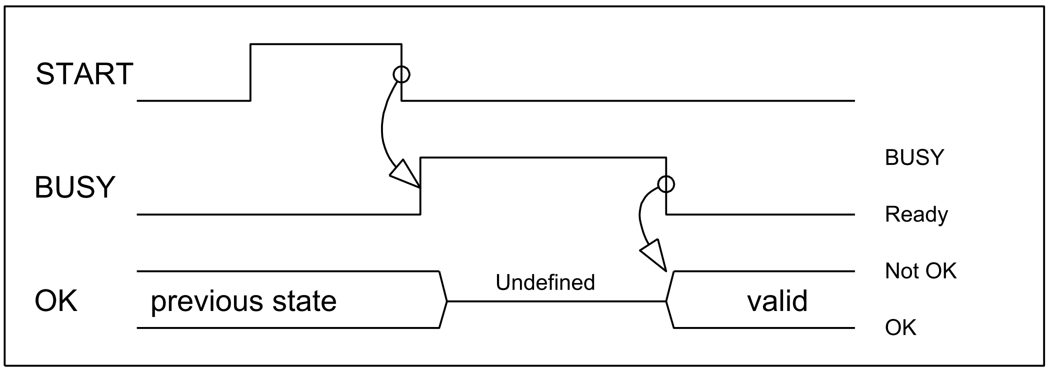
Note

All ways to control Flasher operation work only if Flasher is in standalone mode. In J-Link or MSD mode they have no effect.

5.2 Handshake control

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

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



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

5.3 ASCII command interface

5.3.1 Introduction

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

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

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

5.3.2 General command and reply message format

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

5.3.3 General usage

Reply messages must be considered in each case. In general, a new command must not be sent before a reply for the last one has been received.

5.3.4 Settings for ASCII interface via RS232

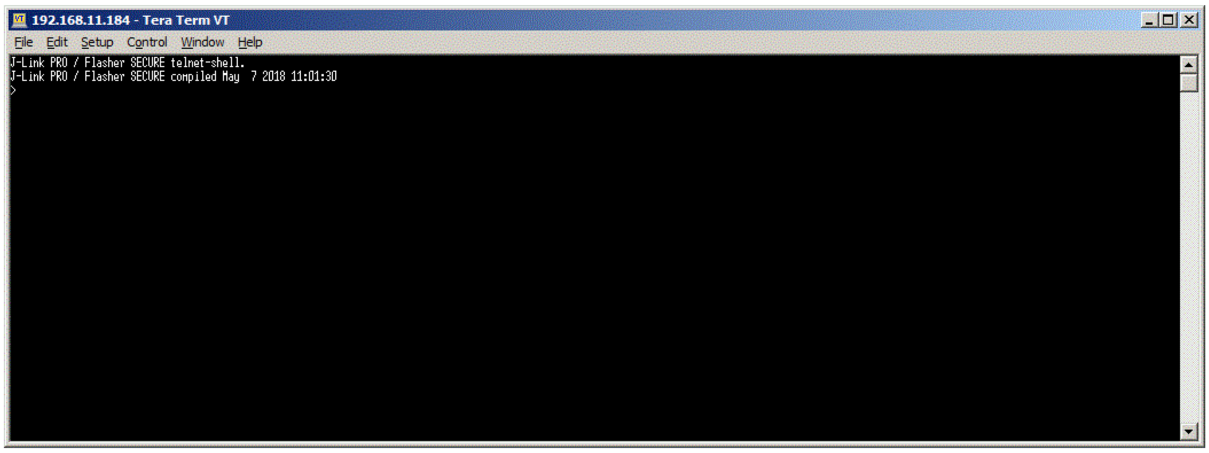
Flasher is driven via a RS232 serial port with the following interface settings:

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

The baud rate can be changed by using the #BAUDRATE command. (see *#BAUDRATE* on page 59)

5.3.5 Settings for ASCII interface via Telnet

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


192.168.11.184 - Tera Term VT
File Edit Setup Control Window Help

```
J-Link PRO / Flasher SECURE telnet-shell.
J-Link PRO / Flasher SECURE compiled May  7 2018 11:01:30
>
```

5.3.6 Commands and replies

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

Commands to the Flasher	Meaning
`#BAUDRATE<Baudrate>`	sets the baud rate of the interface
`#AUTO`	programs target device
`#AUTO PATCH`	patches the binary file and programs target device
`#AUTO NOINFO`	programs target device with less output messages
`#CANCEL`	cancels a command
`#ERASE`	erases the memory
`#PROGRAM`	programs the memory
`#RESULT`	requests the result of the last action
`#SELECT <Filename>`	selects the active project
`#START`	starts the device
`#STATUS`	requests the current status of the Flasher unit
`#VERIFY`	verifies the memory content
File I/O commands	
`#FCLOSE`	closes a file
`#FCRC`	calculates the CRC for a file
`#FDELETE <Filename>`	deletes a file
`#FOPEN <Filename>`	opens a file
`#FREAD <Offset>,<NumBytes>`	reads a file
`#FSIZE`	requests the file size
`#FWRITE <Offset>,<Num-Bytes>:<Data>`	writes a file
`#FLIST`	lists the files
`#MKDIR <Dirname>`	creates a directory
Replies from the Flasher	
`#ACK`	answer command acknowledged
`#NACK`	answer command not acknowledged
`#OK`	answer action finished with status OK
`#OK:<NumBytes>:<Data>`	answer requested bytes
`#OK:<Size>`	answer requested size
`#STATUS:`	answer requested status
`#ERRxxx`	answer action finished with error

Note

Not all commands are supported by all Flasher Types.

5.3.6.1 Commands to the Flasher

5.3.6.1.1 #AUTO

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

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

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

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

Finally, Flasher responds with

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

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

```
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#STATUS:PROGRAMMING
#STATUS:VERIFYING
#OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)
```

5.3.6.1.2 #AUTO PATCH

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

Flasher responds with

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

Note

This feature is not supported by the Flasher Secure.

For further information about the usage of the #AUTO PATCH command please refer to *Patch file support* on page 53.

5.3.6.1.3 #AUTO NOINFO

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

The command ends with #OK or #ERRxxx

5.3.6.1.4 #BAUDRATE<Baudrate>

This command can be sent in order to change the baud rate of the Flasher's RS232 interface used for communication. <Baudrate> is expected in decimal format. The valid range is from 2400 baud to 115.200 baud.

If this command succeeds, Flasher responds with:

```
#ACK
#OK
```

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

```
#ERR255: Invalid parameters
or
#ERR255: Baudrate is not supported
```

Note

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

#CANCEL

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

Flasher will respond with:

```
#ERR007:CANCELED.
```

#ERASE

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

Flasher will reply the following sequence of messages:

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

#PROGRAM

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

#RESULT

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

#SELECT <Filename>

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

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

If this command succeeds, Flasher responds with:

```
#ACK
#OK
```

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

```
#SELECT ATSAM7_1 // ATSAM7_1.CFG and ATSAM7_1.DAT is selected
#ACK
#OK
#AUTO // Start auto programming
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#STATUS:PROGRAMMING
#STATUS:VERIFYING
#OK (Total 8.416s, Erase 0.005s, Prog 6.845s, Verify 0.959s)
#SELECT ATSAM7_2 // ATSAM7_2.CFG and ATSAM7_2.DAT is selected
#ACK
#OK
#AUTO // Start auto programming
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#STATUS:PROGRAMMING
#STATUS:VERIFYING
#OK (Total 8.632s, Erase 0.005s, Prog 7.051s, Verify 0.969s)
```

#START

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

Flasher will reply with the following sequence of messages:

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

#STATUS

This command can be sent any time, even during other command execution. Flasher responds with its current state. All defined state messages are described under *Replies from Flasher* on page 64.

#VERIFY

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

5.3.6.2 File I/O commands

The ASCII interface of the Flasher also supports file I/O operations.

The following file I/O commands are supported:

#FCLOSE

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

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

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

Note

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

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

#FCRC

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

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

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

#FDELETE <Filename>

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

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

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

Note

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

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

#FOPEN <Filename>

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

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

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

Note

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

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

#FREAD <Offset>,<NumBytes>

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

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

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

If the #FREAD command succeeds, Flasher will finally respond with a #OK:<NumBytes>:<Data> reply message. For more information about the Flasher reply messages, please refer to *Replies from Flasher* on page 64.

Note

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

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

#FSIZE

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

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

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

If the #FSIZE command succeeds, Flasher will respond with a #OK:<Size> reply message. For more information about the Flasher reply messages, please refer to *Replies from Flasher* on page 64.

Note

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

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

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

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

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

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

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

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

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

> **Note**
>
> In order to use the #FWRITE command a file has to be opened via the #FOPEN command, first. Otherwise Flasher will respond with the following sequence:
>
> ```
> #ACK
> #ERR255:No file opened
> ```

#FLIST

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

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

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

#MKDIR <Dirname>

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

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

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

> **Note**
>
> If the directory can not be created because of a bad <Dirname> argument, Flasher will respond with:
>
> ```
> #ACK
> #ERR255:Failed to create directory
> ```

5.3.6.3 Replies from Flasher

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

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

The following reply messages from Flasher are defined:

#ACK

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

#NACK

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

#OK

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

#OK:<NumBytes>:<Data>

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

#OK:<Size>

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

#STATUS:

The Flasher replies with its current state.

The following status messages are currently defined:

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

#ERRxxx

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

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

For example:

`#ERR007:CANCELED.`

The error code numbers are described in the following table:

Message	Description
`#ERR007`	Flasher received `#CANCEL` command and has canceled the current operation.
`#ERR008`	Flasher is already busy with execution of previous command.
`#ERR009`	Failed to allocate memory.

Message	Description
`#ERR010`	Failed to open file.
`#ERR011`	Failed to read file.
`#ERR012`	Failed to write file.
`#ERR013`	Failed to delete file.
`#ERR255`	Undefined error occurred. This reply is followed by an error string.

Chapter 6

Device Firmware

The firmware of the target device needs to implement the emSecure software package. The security features of the emSecure software package are used to verify if the flashed device signature is valid or not.

6.1 emSecure Package

The emSecure comes with a detailed documentation itself. Therefore there are only the main topic described in this manual. For the details about the emSecure library take a closer look into the UM12002 emSecure RSA or UM 12004 emSecure ECDSA manual.

6.2 Implementing emSecure

The the emSecure software package is part of the Flasher Secure shipment.

6.2.1 Package content:

emSecure is provided in source code and contains everything needed. The following table shows the content of the emSecure package:

Files	Description
Application	sample applications for bare metal and embOS.
Config	Configuration header files.
Doc	emSecure documentation.
CRYPTO	Shared cryptographic library source code.
SECURE	emSecure implementation code.
SEGGER	SEGGER software component source code used in emSecure.
Sample/Config	Example emSecure configuration.
Sample/Keys	Example emSecure key pairs.
Windows	Supporting applications in binary and source form.

6.2.2 Include directories

You should make sure that the include path contains the following directories (the order of inclusion is of no importance):

- Config
- CRYPTO\Inc
- SECURE\Inc
- SEGGER
- SETUP

You should add the source files from the following directories to your project:

- CRYPTO
- SECURE
- SEGGER
- SETUP\Crypto

Note

Always make sure that you have only one version of each file!

It is frequently a major problem when updating to a new version of emSecure-RSA if you have old files included and therefore mix different versions. If you keep emSecure-RSA in the directories as suggested (and only in these), this type of problem cannot occur. When updating to a newer version, you should be able to keep your configuration files and leave them unchanged. For safety reasons, we recommend backing up (or at least renaming) the SECURE directories before updating.

6.3 Sample Application

The Flasher Secure is shipped with a sample application which looks like the listed example below.

> **Note**
>
> The sample application is build based on the embOS OS. emSecure library can be used without

```
/*********************************************************************
*                SEGGER Microcontroller GmbH & Co. KG                *
*                        The Embedded Experts                        *
**********************************************************************
*                                                                    *
*       (c) 1995 - 2018 SEGGER Microcontroller GmbH & Co. KG         *
*                                                                    *
*       www.segger.com     Support: support@segger.com               *
*                                                                    *
**********************************************************************
*                                                                    *
*       embOS * Real time operating system for microcontrollers      *
*                                                                    *
*                                                                    *
*       Please note:                                                 *
*                                                                    *
*       Knowledge of this file may under no circumstances            *
*       be used to write a similar product or a real-time            *
*       operating system for in-house use.                           *
*                                                                    *
*       Thank you for your fairness !                                *
*                                                                    *
**********************************************************************

----------------------------------------------------------------------
File    : FlasherSecureVerifyExample.c
Purpose : Example code for UID signature check.
--------- END-OF-HEADER ----------------------------------------------
*/

#include "RTOS.h"
#include "BSP.h"
#include "stm32l0xx.h"

#include "SECURE_RSA.h"
#include "SECURE_ECDSA.h"

#include "RSA_PrivateKey.h"
#include "RSA_PublicKey.h"

#include "ECDSA_PrivateKey.h"
#include "ECDSA_PublicKey.h"

int printf(const char *fmt,...);

/*********************************************************************
*
*       Defines, fixed
*
**********************************************************************
*/

//
// Unique device ID register (96 bits)
//
```

```
typedef struct {
  volatile U32 UID_0;
  volatile U32 UID_1;
  volatile U32 UID_2;
} UID_Type;

#define UID ((UID_Type *)0x1FF80050)

/*********************************************************************
*
*       Defines, configurable
*
**********************************************************************
*/

// choose the used cryptographic methode here!
//#define ECDSA                      1   // Use ECDSA
#define RSA                        1   // Use RSA

/*********************************************************************
*
*       Static data
*
**********************************************************************
*/

static OS_STACKPTR int StackHP[128];          /* task stack */
static OS_TASK         TCBHP;                 /* task control block */

static const volatile U8
  __attribute__((section (".otp.SIG"))) _aSignature[256] = { [0 ... 255] = 0xFF };

static const volatile U64 __attribute__((section (".otp.SN")))
  _SerialNo = 0xFFFFFFFFFFFFFFFF;

/*********************************************************************
*
*       Local functions
*
**********************************************************************
*/

static void _HexDump(const U8 *Data, int NumBytes) {
  int i;
  for (i = 0; i < NumBytes; i++) {
    if (!(i % 16)) {
      printf("\n");
    } else if (!(i % 4)) {
      //printf(" ");
    }
    //printf ("%.2X", *Data);
    printf ("0x%.2X, ", *Data);
    Data++;
  }
  printf("\n");
}

#ifdef RSA
static int _VerifyRSA(unsigned char *ID, int NumBytes, int SigBytes) {
  return SECURE_RSA_Verify(&_RSAPublicKey, 0, 0, ID, NumBytes, (U8*)_aSignature, SigBytes);
}
#endif
#ifdef ECDSA
static int _VerifyECDSA(unsigned char *ID, int NumBytes, int SigBytes) {
  SECURE_ECDSA_PUBLIC_KEY PublicKey = { _ECDSAPublicKey, &SECURE_ECDSA_CURVE_P192 };
  return SECURE_ECDSA_Verify(&PublicKey, ID, NumBytes, (U8*)_aSignature, SigBytes);
}
#endif
```

```
/*********************************************************************
*
*       HPTask
*/
static void HPTask(void) {
  while (1) {
    BSP_ToggleLED(0);
    OS_Delay (50);
  }
}

/*********************************************************************
*
*       Global functions
*
**********************************************************************
*/

/*********************************************************************
*
*       MainTask()
*/
#ifdef __cplusplus
extern "C" {     /* Make sure we have C-declarations in C++ programs */
#endif
void MainTask(void);
#ifdef __cplusplus
}
#endif
void MainTask(void) {
  char FW_OK = 0; // check result
  U8   ID[16];    // 16 Bytes UID
  U32  t0;        // time measuring

  t0 = OS_GetTime();
  printf("Unique device ID:\n%.8X %.8X %.8X\n", UID->UID_0, UID->UID_1, UID-
>UID_2);
  SEGGER_memcpy(&ID[0], (U8*)UID, 12);
  memset(&ID[12], 0, 4); // zero padding

  printf("Combined device ID:");
  _HexDump(ID, sizeof(ID));
  printf("\n");

  printf("Serial number: %lld", _SerialNo);
  printf("\n");
#ifdef RSA
  if (_VerifyRSA(ID, sizeof(ID), 256) > 0) {
    printf("Verified UID is correctly signed... SUCCESS!
 (%dms)\n", OS_GetTime() - t0);
    FW_OK = 1;
  } else {
    printf("Signed UID did not verify... ERROR! (%dms)\n", OS_GetTime() - t0);
  }
  printf("\nSignature:");
  _HexDump((U8*)_aSignature, 256);
  printf("\n");
#endif
#ifdef ECDSA
  if (_VerifyECDSA(ID, sizeof(ID), 48) > 0) {
    printf("Verified UID is correctly signed... SUCCESS!
 (%dms)\n", OS_GetTime() - t0);
    FW_OK = 2;
  } else {
    printf("Signed UID did not verify... ERROR! (%dms)\n", OS_GetTime() - t0);
  }
  printf("\nSignature:");
  _HexDump((U8*)_aSignature, 48);
  printf("End.\n", OS_GetTime() - t0);
#endif
  if (FW_OK) {
    //
    // Firmware main task
    //
    OS_CREATETASK(&TCBHP, "HPTask", HPTask, 150, StackHP);
  }
  OS_TerminateTask(NULL);
}

/*************************** End of file ****************************/
```

Features of the example:

- initializes the hardware required for the example,
- reads the unique ID of the device and
- verifies the signature of the device.

6.3.1 The main function

The main function calls the required functions to get the unique ID, verify it and react to the result. Depending on the choosen cryptographic methode the _VerifyRSA or the _VerifyECDSA function is called. The `FW_OK`

variable contains the verification result and the firmware main functions, here repesented by the HPTask, are started is it is set to a value greater then 1. Otherwise the signature check failed and the firmware is not started.

6.3.2 The _VerifyRSA function

If the RSA methode is choosen as cryptographic methode this function will check the signature of the given UID. It calls the corresponding emSecure library function and will report the result.

6.3.3 The _VerifyECDSA function

If the ECDSA methode is choosen as cryptographic methode this function will check the signature of the given UID. It calls the corresponding emSecure library function and will report the result.

Note

The Flasher Secure software package includes either the RSA or the ECDSA cryptographic library.

Chapter 7

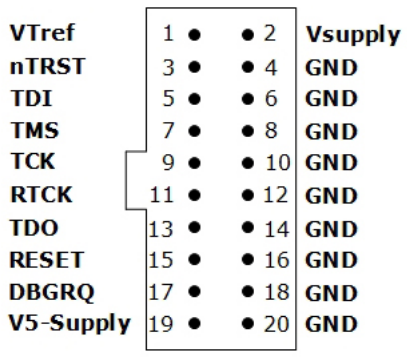
Hardware and Adapters

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

7.1 ARM 20-pin JTAG/SWD Connector

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

7.1.1 Pinout JTAG



The following table lists the JTAG pinout.

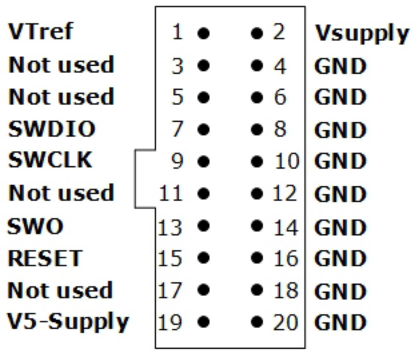
PIN	SIGNAL	TYPE	Description
1	VTref	Input	This is the target reference voltage. It is used to check if the target has power, to create the logic-level reference for the input comparators and to control the output logic levels to the target. It is normally fed from Vdd of the target board and must not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

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

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

7.1.2 Pinout SWD

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



The following table lists the SWD pinout.

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

PIN	SIGNAL	TYPE	Description
15	RESET	I/O	Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called "nRST", "nRESET" or "RESET".
17	Not used	NC	This pin is not connected in Flasher Secure.
19	5V-Target supply	Output	This pin is used to supply power to some eval boards. Not all J-Links supply power on this pin, only the KS (Kickstart) versions. Typically left open on target hardware.

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

7.1.3 Target power supply

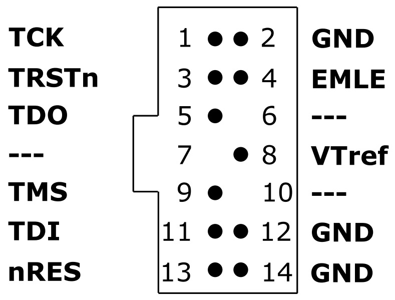
Pin 19 of the connector can be used to supply power to the target hardware. Supply voltage is 5V, max. current is 400mA. The output current is monitored and protected against overload and short-circuit.

Power can be controlled via the J-Link commander. The following commands are available to control power:

Command	Explanation
`power on`	Switch target power on
`power off`	Switch target power off
`power on perm`	Set target power supply default to "on"
`power off perm`	Set target power supply default to "off"

7.2 Flasher RX 14-pin connector

Flasher Secure itself has a 20-pin JTAG connector mounted but an optional J-Link 14-pin RX adapter is available. This adapter enables Flasher Secure to optionally power the connected target hardware. On the adapter there is a jumper which allows selection between 3.3V and 5V supply target voltage supply. The target is supplied via the VTref connection when the supply option is jumpered.



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

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

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

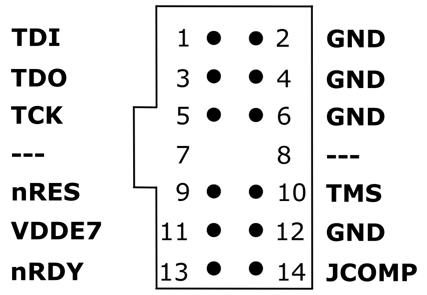
7.2.1 Target power supply

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

Command	Explanation
`power on`	Switch target power on
`power off`	Switch target power off
`power on perm`	Set target power supply default to “on”
`power off perm`	Set target power supply default to “off”

7.3 Flasher PPC 14-pin connector

Flasher Secure itself has a 20-pin JTAG connector mounted but an optional 14-pin PPC adapter for PowerPC devices is available.



The following table lists the J-Link PPC 14-pin adapter JTAG pinout.

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

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

7.4 Target board design

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

- *Pinout JTAG* on page 75
- *Pinout SWD* on page 76
- *J-Link RX 14-Pin Adapter, J-Link RX FINE 14-Pin Adapter* on page 78
- *J-Link PPC 14-Pin Adapter* on page 80

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

7.4.1 Pull-up/pull-down resistors

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

7.4.2 RESET, nTRST

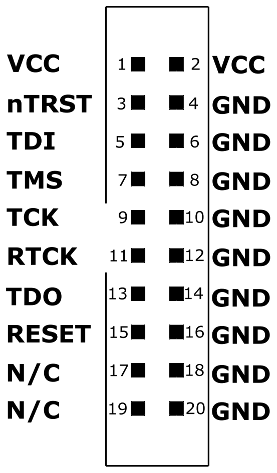
The debug logic is reset independently from the CPU core with nTRST. For the core to operate correctly it is essential that both signals are asserted after power-up.

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

7.5 Adapters

7.5.1 JTAG Isolator

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



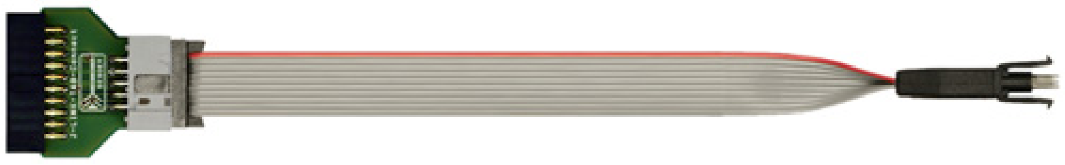
7.5.1.1 Pinout

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

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

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

7.5.2 J-Link Needle Adapter



Why to choose the J-Link Needle Adapter:

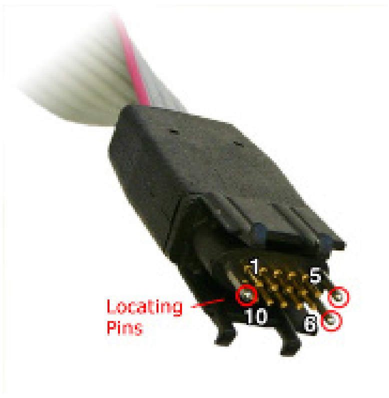
1. No additional connector required on your PCB
2. Very small footprint
3. High reliability spring pins for secure connections
4. Designed with 3 locating pins, so the adapter can not be connected the wrong way
5. No external power supply required! The J-Link Needle Adapter comes with the option to power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

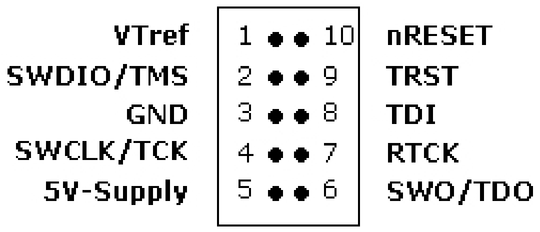
These features make the J-Link Needle Adapter the perfect solution for production purposes.

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

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

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





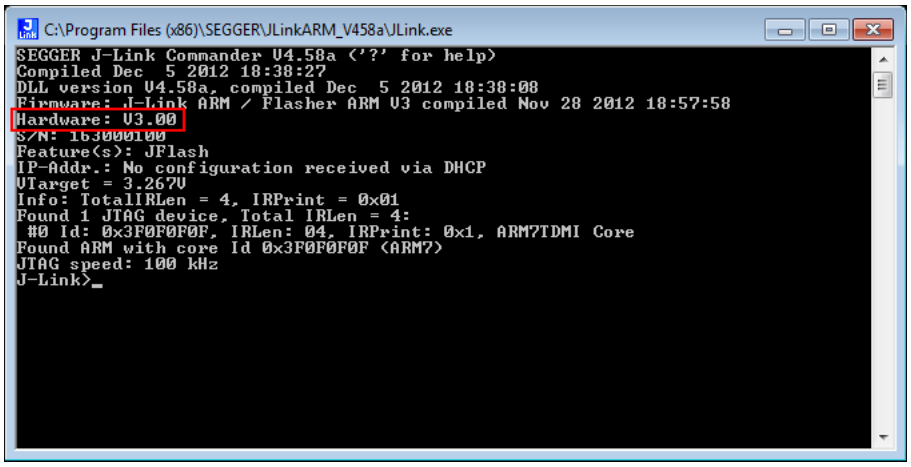
VTref	1	● ●	10	nRESET
SWDIO/TMS	2	● ●	9	TRST
GND	3	● ●	8	TDI
SWCLK/TCK	4	● ●	7	RTCK
5V-Supply	5	● ●	6	SWO/TDO

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

7.6 How to determine the hardware version

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

If this is not the case with your Flasher, you can use `JLink.exe` to determine your hardware version (if Flasher is in J-Link mode). As part of the initial message, the hardware version is displayed. For more information about how to ensure that Flasher is in J-Link mode, please refer to *J-Link mode* on page 53.



C:\Program Files (x86)\SEGGER\JLinkARM_V458a\JLink.exe

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

Chapter 8

Support

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

8.1 Contacting support

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

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

- A detailed description of the problem
- the serial number
- Information about your target hardware (processor, board, etc.).
- `FLASHER.CFG`, `FLASHER.DAT`, `FLASHER.LOG`, `SERIAL.TXT` file from the Flasher. To get these files, Flasher has to be in MSD mode. For more information about how to boot the unit in MSD mode, please refer to *MSD mode* on page 46. In case of a Flasher Secure some files need to be shared from the Flasher Secure Server.

Flasher Secure is sold directly by SEGGER.

Chapter 9

Glossary

This chapter describes important terms used throughout this manual.

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

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

CM
Contract Manufacturer: A manufacturer that contracts with the customer for components or products.

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

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

Halfword
A 16-bit unit of information.

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

ICache
Instruction cache.

ID
Identifier.

IEEE 1149.1
The IEEE Standard which defines TAP. Commonly (but incorrectly) referred to as JTAG.

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

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

IR
See Instruction Register.

JTAG
Joint Test Action Group: The name of the standards group which created the IEEE 1149.1 specification.

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

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

MMU
Memory management unit: Hardware that controls caches and access permissions to blocks of memory, and translates virtual to physical addresses.

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

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

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

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

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

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

RTOS
Real Time Operating System.

TAP
Test Access Port: The port used to access a device’s TAP Controller. Comprises TCK, TMS, TDI, TDO, and nTRST (optional).

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

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

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

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

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

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

UID
Unique Identifier: A number (typically about 128bit) chosen by the chip vendor during production of the chip. This number is unique per device and cannot be changed.

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

Chapter 10

Literature and references

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

Reference	Title	Comments
[J-Link]	J-Link / J-Trace User Guide	This document describes J-Link and J-Trace. It is publicly available from SEGGER *https://www.segger.com/downloads/jlink/UM08001*.
[J-Flash]	J-Flash User Guide	This document describes J-Flash. It is publicly available from SEGGER *https://www.segger.com/downloads/jlink/UM08003*.