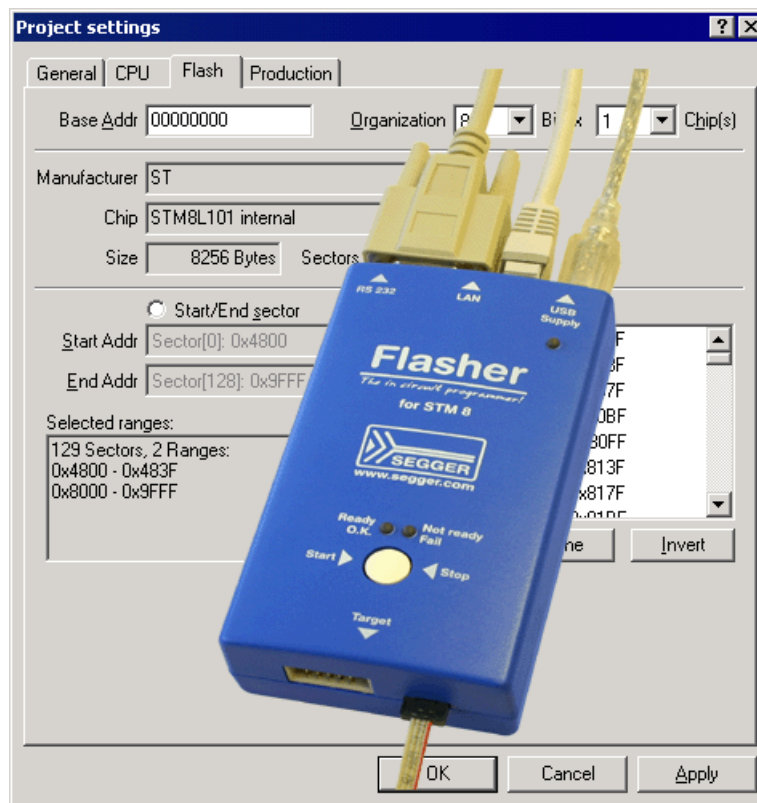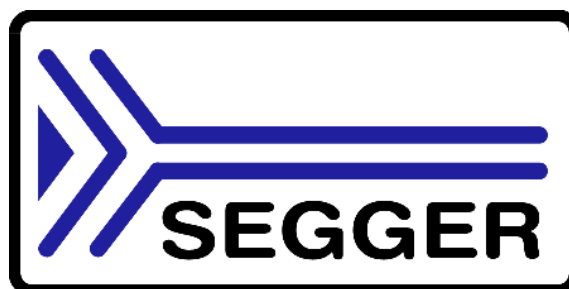# *Flasher STM8*

## User guide of the stand-alone
## SWIM programmer for STM8 Cores



## Manual Rev. 15

**Date: March 2, 2020**

## Document: UM05006



**A product of SEGGER Microcontroller GmbH**

**www.segger.com**

**Disclaimer**

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

**Copyright notice**

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

© 2020 SEGGER Microcontroller GmbH, Hilden / Germany

**Trademarks**

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

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

**Contact address**

SEGGER Microcontroller GmbH

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

Tel.+49 2173-99312-0
Fax.+49 2173-99312-28
Email: ticket_flasher@segger.com
Internet: *https://www.segger.com*

## Revisions

This manual describes the Flasher STM8 device.

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

| Revision | Date | By | Explanation |
|---|---|---|---|
| 15 | 200302 | OO | Chapter "Remote control":<br>- Added command "#POWER" |
| 14 | 180515 | OO | Updated company name.<br>Chapter "Remote control":<br>- Added command "#FCRC" |
| 13 | 160819 | OO | Chapter "Remote control":<br>- Added command "#FLIST"<br>- Added command "#FORMATHIGH"<br>- Added command "#FORMATLOW" |
| 12 | 150410 | OO | Chapter "Remote control":<br>- Added command "#READMEM". |
| 11 | 150318 | OO | Chapter "Remote control":<br>- Removed "obsolete" warning for 10-pin connector.<br>- Added information regarding SWIM pull-up. |
| 10 | 130404 | OO | Chapter "Remote control":<br>- Added ASCII interface command "#SELECT". |
| 9 | 110811 | OO | Some minor changes. |
| 8 | 110502 | OO | Chapter "Hardware":<br>- Some minor changes.<br>- Added more detailed information about the connection cable. |
| 7 | 110307 | OO | Chapter "Introduction":<br>- Added speed measurements |
| 6 | 101007 | OO | Chapter "Working with Flasher STM8":<br>- Added example for setting up multiple configurations for stand-alone mode and how to switch between configurations remotely. |
| 5 | 100702 | OO | Chapter "Remote control":<br>- Added information regarding TCP/IP terminal |
| 4 | 100624 | OO | Updated document to renamed binaries in Flasher STM8 software and documentation package. Updated all screenshots. |
| 3 | 100607 | OO | Chapter "Hardware":<br>- Added information about connecting Flasher GND to power supply ground for more power stabilization |
| 2 | 100525 | OO | Chapter "Hardware":<br>- Added more information to SWIM and Reset output signals<br>- Some minor changes |
| 1 | 100401 | OO | Chapter "Working with Flasher STM8":<br>- Added information about setup multiple Flasher on one PC<br>Chapter "Hardware":<br>- Declared 10-pin connector as obsolete interface |
| 0 | 091210 | OO | Initial version. |

# About this document

This document describes the Flasher STM8. It provides an overview over the major features of the Flasher STM8, gives you some background information about SWIM, STM8 general and describes Flasher STM8 related software packages available from Segger. Finally, the chapter *Support and FAQs* on page 67 helps to troubleshoot common problems.

## Typographic conventions

This manual uses the following typographic conventions:

| Style | Used for |
|---|---|
| Body | Body text. |
| Keyword | Text that you enter at the command-prompt or that appears on the display (that is system functions, file- or pathnames). |
| *Reference* | Reference to chapters, tables and figures or other documents. |
| **GUIElement** | Buttons, dialog boxes, menu names, menu commands. |

**Table 1.1: Typographic conventions**

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

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

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

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

**Corporate Office:**
*http://www.segger.com*

**United States Office:**
*http://www.segger-us.com*

# EMBEDDED SOFTWARE (Middleware)

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

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

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

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

# SEGGER TOOLS

### Flasher
**Flash programmer**
Flash Programming tool primarily for microcontrollers.

### J-Link
**JTAG emulator for ARM cores**
USB driven JTAG interface for ARM cores.

### J-Trace
**JTAG emulator with trace**
USB driven JTAG interface for ARM cores with Trace memory. supporting the ARM ETM (Embedded Trace Macrocell).

### J-Link / J-Trace Related Software
Add-on software to be used with SEGGER's industry standard JTAG emulator, this includes flash programming software and flash breakpoints.

# Table of Contents

# Chapter 1

# Introduction

This chapter gives a short overview about the Flasher STM8.

# 1.1    Flasher STM8 overview

Flasher STM8 is a programming tool for microcontrollers with on-chip Flash memory and STM8 core. Flasher STM8 is designed for programming flash targets with a PC program or stand-alone.

Flasher STM8 connects via USB, via ethernet or via RS232 interface to a PC, running Microsoft Windows 2000, Windows XP, Windows 2003, Windows Vista or Windows 7. Flasher STM8 has a built-in 10-pin interface connector and a built-in 4-pin interface connector, which are compatible with the debug connectors used by STM8 eval systems.

## 1.1.1    Features of Flasher STM8

- USB 2.0 interface
- Full duplex 100Mbit ethernet interface
- TCP/IP server included in Flasher STM8, which allows using Flasher via TCP/IP networks
- Standard 4-pin SWIM connector
- USB, ethernet, RS232 and 4-pin ribbon cable included
- No power supply required, powered through USB
- Target power supply via pin 1 of the 4-pin interface (up to 300mA to target with overload protection)
- Integrated optical isolation between host and target system
- Support for all STM8 devices
- Target voltage can be measured
- Fully plug and play compatible
- Three boot modes: PC mode, stand-alone mode, MSD mode
- Stand-alone SWIM programmer (Once set up, Flasher can be controlled without the use of PC program)
- 128 MB memory for storage of target program
- Serial in target programming supported
- Data files can be updated via PC program
- Flash programming software available

## 1.1.2    Working environment

### General

Flasher STM8 can be operated from a PC with an appropriate software like Flasher STM8 software or in stand-alone mode.

### Host System

IBM PC/AT or compatible. CPU: Pentium with at least 192MB of RAM, running Microsoft Windows 2000, Windows XP, Windows 2003, Windows Vista or Windows 7. It needs to have an USB, ethernet or RS232 interface available for communication with Flasher STM8.

### Power supply

Flasher requires 5V DC, min. 100mA via USB connector. If USB is not connected, the USB connector is used to power the device. Supply voltage is the same in this case. Please avoid excess voltage.

### Installing Flasher STM8 PC-software Flasher STM8

The latest version of the Flasher STM8 Software, which is part of the Flasher STM8 software and documentation package, can be downloaded from our website: *http://www.segger.com*. For more information about using Flasher STM8 Software please refer to the *Flasher STM8 Software User Guide* which is also available for download on our website.

# 1.2 Specifications

## 1.2.1 Specifications for Flasher STM8

The following table gives an overview about the specifications (general, mechanical, electrical) for Flasher STM8.

| General | |
|---|---|
| Supported OS | Microsoft Windows 2000<br>Microsoft Windows XP<br>Microsoft Windows XP x64<br>Microsoft Windows 2003<br>Microsoft Windows 2003 x64<br>Microsoft Windows Vista<br>Microsoft Windows Vista x64<br>Windows 7<br>Windows 7 x64 |
| Electromagnetic compatibility (EMC) | EN 55022, EN 55024 |
| Operating temperature | +5°C ... +60°C |
| Storage temperature | -20°C ... +65 °C |
| Relative humidity (non-condensing) | Max. 90% rH |
| Size (without cables) | 121mm x 66mm x 30mm |
| Weight (without cables) | 122g |
| **Mechanical** | |
| USB interface | USB 2.0, full speed |
| Ethernet interface | 100MHz full duplex |
| RS232 Host Interface | RS232 9-pin |
| Target interface | SWIM (either via 10-pin connector or 4-pin connector depending on target hardware) |
| **SWIM Interface, Electrical** | |
| Power supply | USB powered, 100mA for Flasher STM8. 500mA if target is powered by Flasher STM8. |
| Target interface voltage ($V_{IF}$) | 1.2V ... 5V |
| Target supply voltage | 4.5V ... 5V (if powered with 5V on USB) |
| Target supply current | Max. 300mA |
| LOW level input voltage ($V_{IL}$) | Max. 40% of $V_{IF}$ |
| HIGH level input voltage ($V_{IH}$) | Min. 60% of $V_{IF}$ |

**Table 1.1: Flasher STM8 specifications**

# 1.3    Supported CPU cores

Flasher STM8 has been designed and tested with the following cores, but should work with any STM8 core. If you experience problems with a particular core, do not hesitate to contact Segger.

- STM8A
- STM8L
- STM8S

# 1.4　Download speed

Flasher STM8 has been designed to allow downloading to the target as fast as the target can receive and program flash. The table below shows the measured download speed for standalone programming and for programming via PC driven software. The following testing environment has been used:

- PC with 2.53 GHz Core2Duo, running WinXP
- USB 2.0 port
- USB 2.0 hub
- Flasher STM8
- STM8S208MBT6B

| Operation | Standalone mode | PC mode |
|---|---|---|
| Program (128kB) | 8.5s (15kB/s) | 8.5s (15kB) |
| Program and verify (128kB) | 11.3s | 11.3s |

**Table 1.2: Flasher STM8 download speed**

# Chapter 2

# Setup

This chapter describes the setup procedure required in order to work with Flasher STM8. Primarily this includes the installation of the Flasher STM8 software and documentation package.

# 2.1    Installing the Flasher STM8 software and documentation pack

Flasher STM8 is shipped with a bundle of applications, corresponding manuals and some sample projects and the kernel mode USB driver.

Refer to chapter *Flasher STM8 related software* on page 27 for an overview about the Flasher STM8 software and documentation pack.

## 2.1.1    Setup procedure

To install the Flasher STM8 software and documentation pack, follow this procedure:

**Note:**      We recommend to check if a newer version of the Flasher STM8 software and documentation pack is available for download before starting the installation. Check therefore the Flasher STM8 related download section of our website.

Before you plug your Flasher STM8 into your computer's USB port, extract the setup tool `Setup_FlasherSTM8_V<VersionNumber>.zip`. The setup wizard will install the software and documentation pack that also includes the certified J-Link USB driver. Start the setup by double clicking `Setup_FlasherSTM8_V<VersionNumber>.exe`. The **license Agreement** dialog box will be opened. Accept the terms with the **Yes** button.
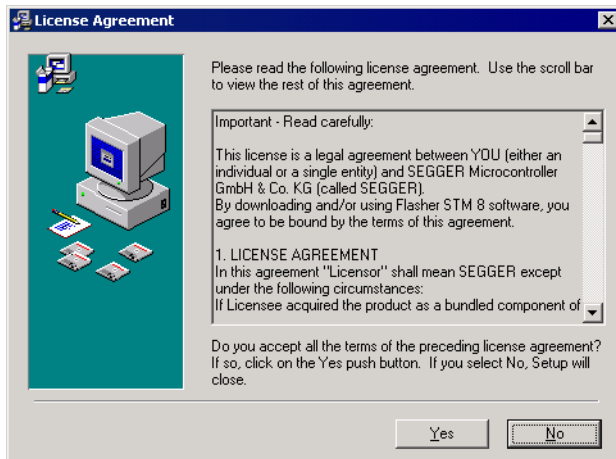


1.    The **Welcome** dialog box is opened. Click **Next >** to open the **Choose Destination Location** dialog box.
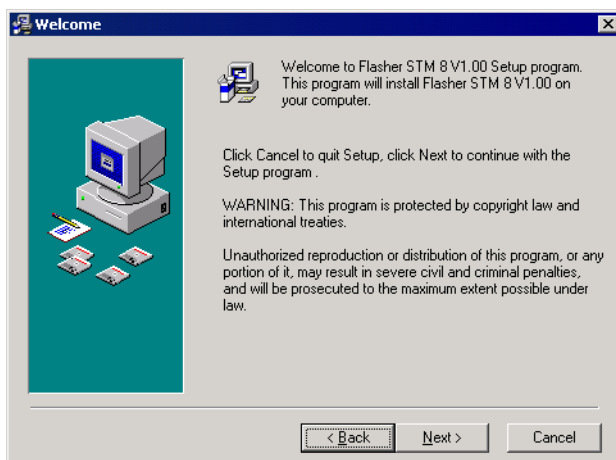
2. Accept the default installation path `C:\Program Files\SEG-GER\FlasherSTM8_V<VersionNumber>` or choose an alternative location. Confirm your choice with the **Next >** button.
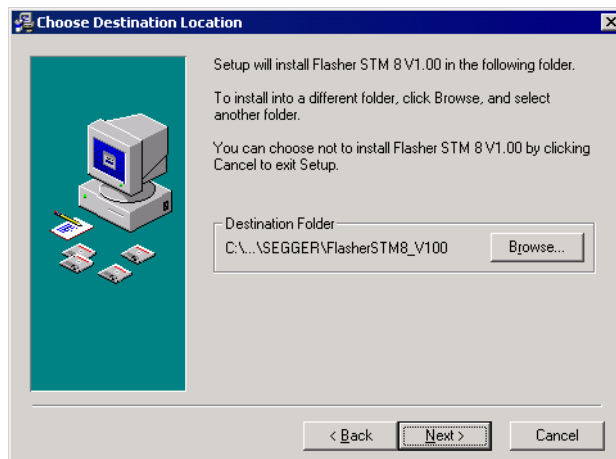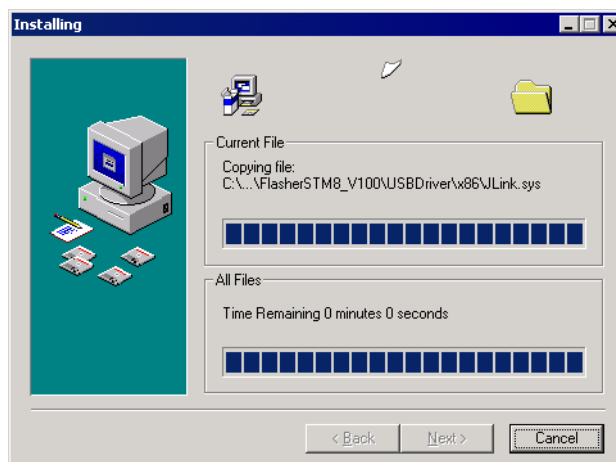


3. The **Choose options** dialog is opened. The **Create entry in start menu** option is preselected. Accept or deselect the options and confirm the selection with the **Next >** button.
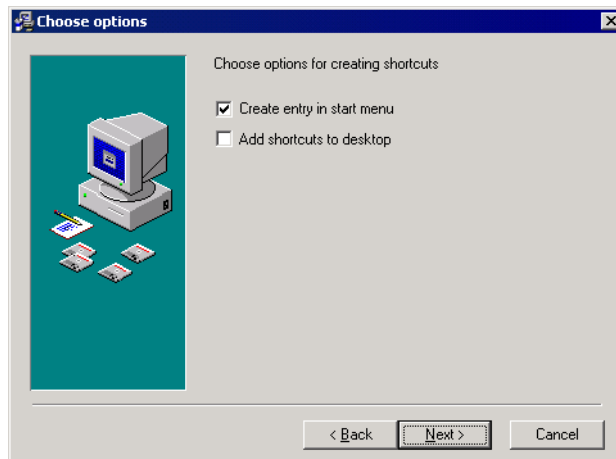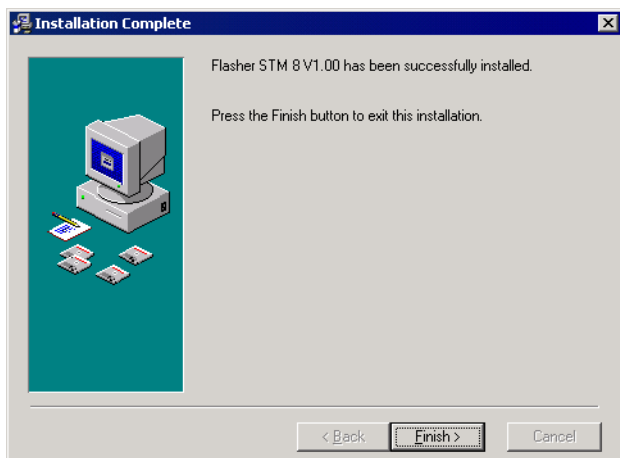


Confirm start of installation by pressing **Next >** button again.

4. The installation process will be started.

5. The **Installation Complete** dialog box appears after the copy process. Close the installation wizard with the **Finish >** button.



The Flasher STM8 software and documentation pack is successfully installed on your PC.

6. Connect your Flasher STM8 via USB with your PC. The Flasher STM8 will be identified and after a short period the Flasher LED stops rapidly flashing and stays on permanently.
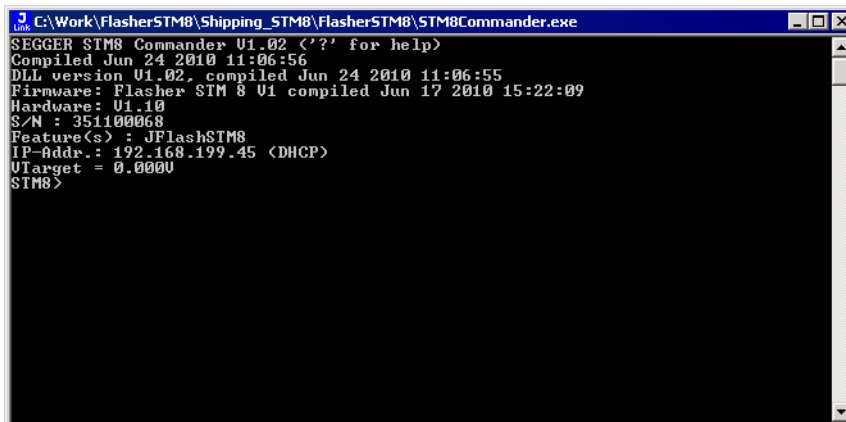
## 2.2 Setting up the USB interface

After installing the Flasher STM8 software and documentation package it should not be necessary to perform any additional setup sequences in order to configure the USB interface of Flasher STM8.

### 2.2.1 Verifying correct driver installation

To verify the correct installation of the driver, disconnect and reconnect Flasher STM8 to the USB port. During the enumeration process which takes about 2 seconds, the LED on Flasher STM8 is flashing. After successful enumeration, the LED stays on permanently.

Start the provided sample application `STM8Commander.exe`, which should display the compilation time of the Flasher firmware, the serial number and a target voltage of 0.000V. The screenshot below shows an example.



In addition you can verify the driver installation by consulting the Windows device manager. If the driver is installed and your Flasher STM8 is connected to your computer, the device manager should list the J-Link USB driver as a node below "Universal Serial Bus controllers" as shown in the following screenshot:
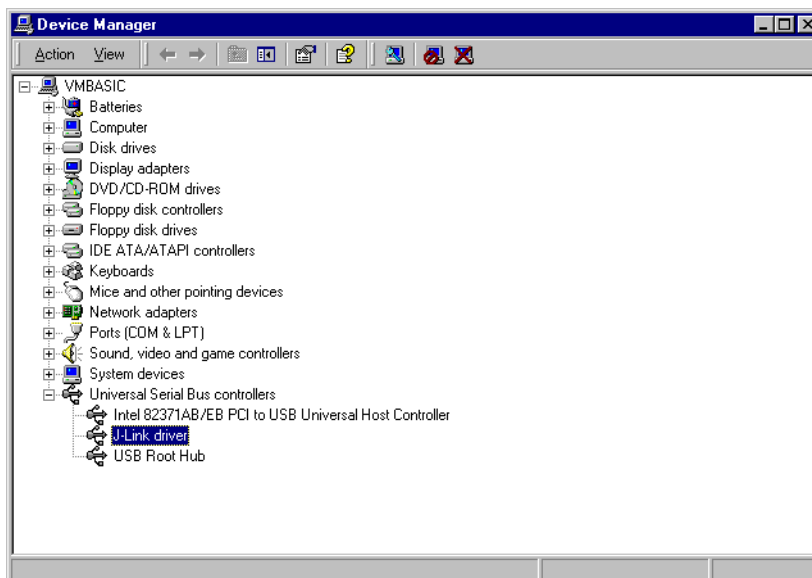
Right-click on the driver to open a context menu which contains the command **Properties**. If you select this command, a **J-Link driver Properties** dialog box is opened and should report: **This device is working properly**.
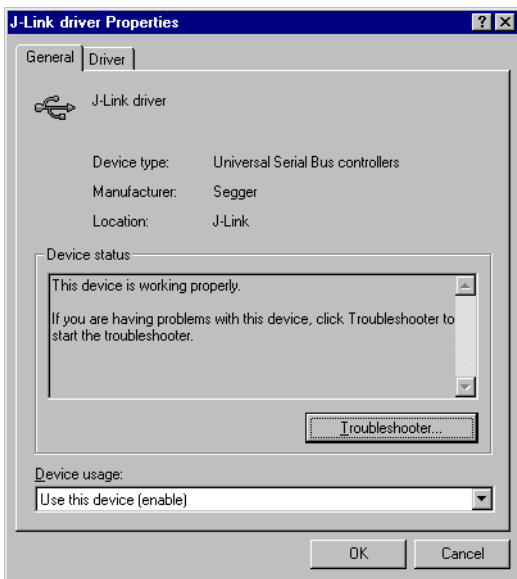
If you experience problems, refer to the chapter *Support and FAQs* on page 67 for help. You can select the **Driver** tab for detailed information about driver provider, version, date and digital signer.

# 2.3 Uninstalling the J-Link USB driver

If Flasher STM8 is not properly recognized by Windows and therefore does not enumerate, it makes sense to uninstall the J-Link USB driver.

This might be the case when:

- The LED on the Flasher STM8 is rapidly flashing.
- The Flasher STM8 is recognized as **Unknown Device** by Windows.

To have a clean system and help Windows to reinstall the J-Link driver, follow this procedure:

1. Disconnect Flasher STM8 from your PC.
2. Open the **Add/Remove Programs** dialog (`Start > Settings > Control Panel > Add/Remove Programs`) and select **Windows Driver Package - Segger (jlink) USB** and click the **Change/Remove** button.
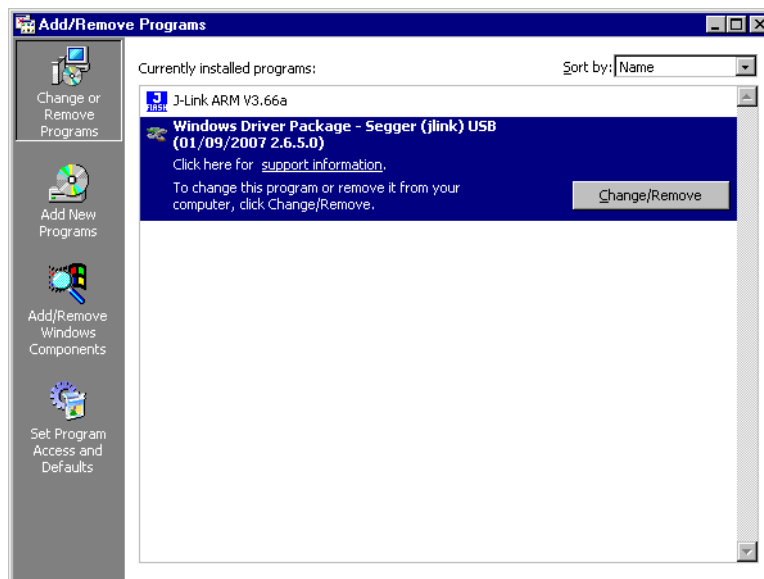


3. Confirm the uninstallation process.
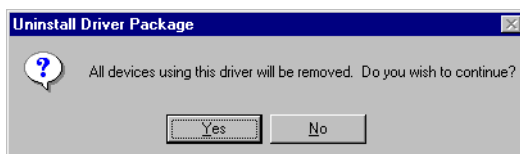
# 2.4    Setting up the IP interface

Flasher STM8 has an additional Ethernet interface, to communicate with the host system. A built-in web server allows configuration of the flasher via web interface. In addition to that, you can set a default gateway for the flasher which allows using it even in large intranets. For simplicity the setup process of Flasher STM8 is described in this section.

## 2.4.1    Connecting the first time

When connecting Flasher the first time, it attempts to acquire an IP address via DHCP. To get information about which IP address is acquired, you have two possibilities:

*   Connecting Flasher via USB and via Ethernet and read out the IP address via `STM8Commander.exe`.
*   Connecting Flasher only via Ethernet and read out the IP via the DHCP IP Assignment table of your DHCP Server.

In the following, both ways to get the IP address assigned to Flasher via DHCP, are explained.

### 2.4.1.1    Connecting via USB and Ethernet

When using `STM8Commander.exe` in order to read out the IP address, Flasher has to be connected to your host system via Ethernet and via USB. When starting `STM8Commander.exe`, it will show information about the IP address (static / dynamic) when connecting to Flasher.



To get more detailed information about the current configuration of the Flasher (such as subnet mask and MAC address), you can use the `conf` command in `STM8Commander.exe`.
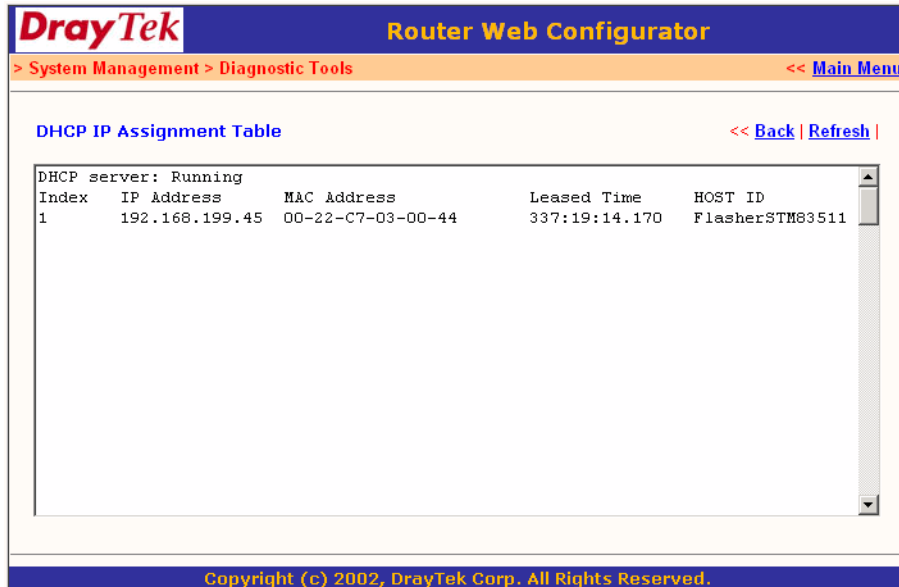


After reading out the IP address you can connect to Flasher via Ethernet, using the IP address.

### 2.4.1.2   Connecting via Ethernet only

This way of reading out the IP address of Flasher can be used for example if you do not have administrator rights on the host system in order to install the USB driver which is necessary to connect to Flasher via USB. To get the IP address which has been assigned to Flasher via DHCP, you have to read it out from the DHCP IP Assignment table of your DHCP Server:



You can easily identify your Flasher by its host ID (in this case FLASHER351100023) and by its MAC addr which always starts with: 00-22-C7-03-XX-XX where XX depends on the last five digits of the serial number of your Flasher. In this case the serial number of the connected Flasher is 351100023 (0x0017), so its MAC address is: 00-22-C7-03-00-17.

## 2.4.2   Configuring the Flasher

By default, Flasher is configured to receive an IP address and a subnet mask via DHCP. It is also possible to assign a fixed IP address to it. Setting up Flasher can be done via `STM8Commander.exe` or via web interface. In the following, both configuration methods are described.

### 2.4.2.1   Configuring Flasher via STM8Commander.exe

Configuring Flasher via `STM8Commander.exe` is very simple because only one command (in different variations) is necessary to choose between automatic IP address and dynamic IP address assignment.

**Note:**     If you want to configure Flasher via STM8Commander.exe and Flasher is connected to your host-system via Ethernet only, you have to type in the `ip <IPAddr>` command.

**Example**

```
ip 192.168.199.29
```

**Assigning an IP address via DHCP**

By default, Flasher is configured to acquire an IP address via DHCP, so it should not be necessary to configure this. But, if you change the IP address to a fixed one, DHCP is disabled from this point. To re-enable DHCP you should use the `ipaddr DHCP` command in `STM8Commander.exe`. The ipaddr command will be explained in the following.

### Assigning an IP address manually

If you do not want Flasher to be configured via DHCP, you can assign an IP address and a subnet mask (optional) manually. This is done via the `ipaddr` command in `STM8Commander.exe`. This command can be used in four different ways, which are explained in the table below:

| Command | Explanation |
|---|---|
| `ipaddr` | If no additional parameter is specified, the current IP and subnet mask of Flasher are shown. |
| `ipaddr <IP>` | If an IP is given as an additional parameter the given IP address is set as the IP address for Flasher. A default 16-bit subnet mask (255.255.0.0) is used. From this time Flasher uses this static IP, DHCP is disabled from this point. |
| `ipaddr <IP> <Subnet mask>` | If an IP and a subnet mask is given as an additional parameter, the given IP and the given subnet mask are used. From this time Flasher uses this static IP and subnet mask, DHCP is disabled from this point. |
| `ipaddr DHCP` | If DHCP is given as an additional parameter the use of DHCP is enabled. Previously made IP settings are discarded. |

**Table 2.1: ipaddr command description**

### Example ipaddr

```
STM8>ipaddr
DHCP assigned network configuration
IP-Addr:    192.168.199.29
Subnetmask: 255.255.0.0
```

### Example ipaddr <IP>

```
STM8>ipaddr 192.168.87.115
IP address successfully changed to '192.168.87.115'.
Subnetmask successfully changed to '255.255.0.0'.
```

### Example ipaddr <IP> <Subnet mask>

```
STM8>ipaddr 192.168.87.116 255.255.0.0
IP address successfully changed to '192.168.87.116'.
Subnetmask successfully changed to '255.255.0.0'.
```

### Example ipaddr DHCP

```
STM8>ipaddr DHCP
Configuration successfully changed to DHCP.
```

## 2.4.2.2   Configuring Flasher via web interface

Flasher comes with a web server, which provides a web interface for configuration. This enables you to configure Flasher without additional tools, just with a simple web browser. The **Home** page of the web interface shows the serial number, the current IP address and the MAC address of the Flasher.



The **Network configuration** page allows you to configure the IP address, the subnet mask and the default gateway of Flasher. You can choose between **automatic** IP assignment and **manual** IP assignment by selecting the appropriate radio button. If you choose **manual**, you can change the IP address, the subnet mask and the default gateway by entering the desired values in the appropriate fields and clicking **change**. So, you do not have to care about any command syntax in order to change the IP address/subnet mask/default gateway.

# Chapter 3

# Flasher STM8 related software

This chapter describes Segger's Flasher STM8 related software portfolio available for use with Flasher STM8 hardware.

# 3.1    Flasher STM8 software and documentation package

Flasher STM8 is shipped with a bundle of applications.

| Software | Description |
|---|---|
| JLinkSTM8.dll | DLL for using Flasher STM8 with third-party programs. |
| STM8Commander.exe | Free command-line tool with basic functionality for target analysis. |
| FlasherSTM8.exe | Stand-alone flash programming application. For more information about Flasher STM8 Software please refer to *Flasher STM8 Software User's Guide (UM05007)*. |
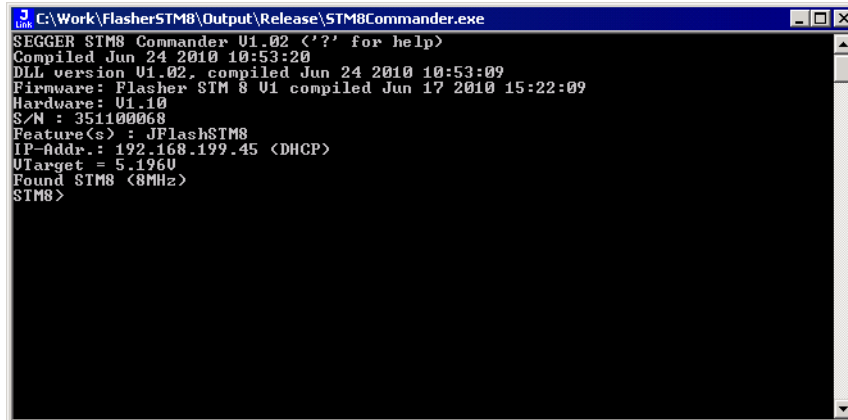
**Table 3.1: Flasher STM8 related software**

## 3.2 Flasher STM8 software and documentation package in detail

The Flasher STM8 software documentation package is shipped together with Flasher STM8 and may also be downloaded from *www.segger.com*.

### 3.2.1 STM8 Commander (Command line tool)

STM8 Commander (`STM8Commander.exe`) is a tool that can be used for verifying proper installation of the USB driver and to verify the connection to the STM8 chip, as well as for simple analysis of the target system. It permits some simple commands, such as memory dump, halt, step and go. You can get an overview of the available commands by simply using the [ENTER] key or "?".

## 3.2.2    Flasher STM8 Software (Program flash memory)

Flasher STM8 is a software running on Windows 2000, Windows XP, Windows 2003, Windows Vista or Windows 7 systems and enables you to program your flash EEPROM devices via the SWIM connector on your target system.

Flasher STM8 works with any STM8 system and supports the programming of internal flash of STM8 microcontrollers. It allows you to erase, fill, program, blank check, upload flash content, and view memory of your flash devices.
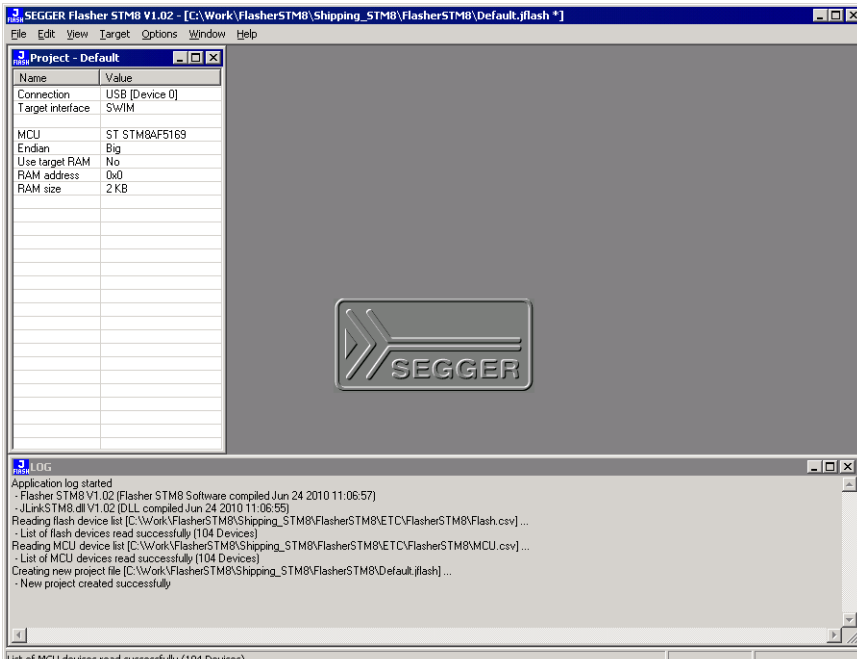


### Features

- Works with any STM8 chip
- STM8 microcontrollers (internal flash) supported
- Smart read-back: Only non-blank portions of flash transferred and saved
- Easy to use, comes with projects for standard eval boards.

# 3.3 Using the JLinkSTM8.dll

## 3.3.1 What is the JLinkSTM8.dll?

The JLinkSTM8.dll is a standard Windows DLL typically used from C or C++, but also Visual Basic or Delphi projects. It makes the entire functionality of the Flasher STM8 available through the exported functions.

The functionality includes things such as halting/running the STM8 core and reading/writing memory. Therefore, it can be used in any kind of application accessing an STM8 core.

## 3.3.2 Determining the version of JLinkSTM8.dll

To determine which version of the JLinkSTM8.dll you are facing, the DLL version can be viewed by right clicking the DLL in explorer and choosing `Properties` from the context menu. Click the `Version` tab to display information about the product version.
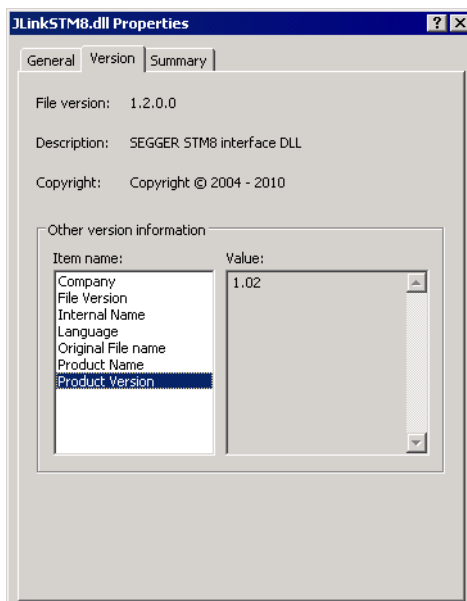
### 3.3.3   Determining which DLL is used by a program

To verify that the program you are working with is using the DLL you expect it to use, you can investigate which DLLs are loaded by your program with tools like Sysinternals' Process Explorer. It shows you details about the DLLs, used by your program, such as manufacturer and version.
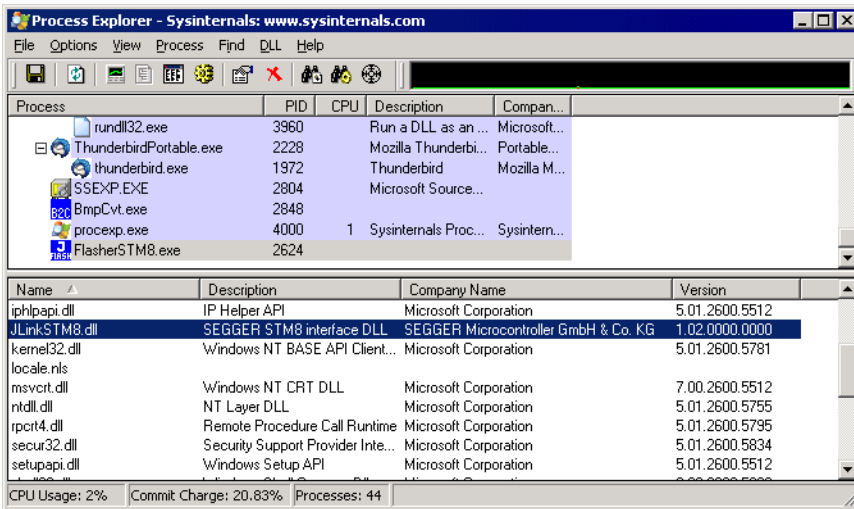


Process Explorer is - at the time of writing - a free utility which can be downloaded from *www.sysinternals.com*.

# Chapter 4

# Working with Flasher STM8

This chapter describes functionality and how to use Flasher STM8.

# 4.1    Operating modes

Flasher STM8 is able to boot in 3 different modes:

*   PC mode
*   Stand-alone mode
*   MSD (Mass storage device) mode

If Flasher STM8 can enumerate on the USB port, Flasher STM8 boots in "PC mode". In this mode Flasher STM8 can be used as an emulator. When Flasher is powered up and Flasher STM8 cannot enumerate, the "stand-alone mode" is started. In this mode Flasher STM8 can be used as a stand-alone flash programmer. When the Start/Stop button is kept pressed when Flasher is powered, Flasher STM8 boots in "MSD mode". In this mode Flasher STM8 boots as a mass storage device.

## 4.1.1    PC mode

When you want to use Flasher STM8 for the first time you need to install the Flasher STM8 related software and documentation pack. After installation, connect Flasher STM8 to the host PC via USB.
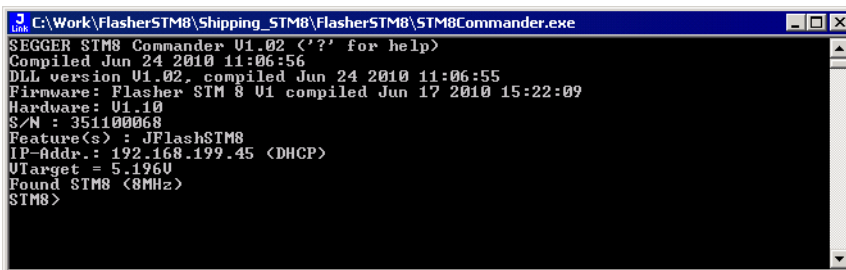
### 4.1.1.1    Connecting the target system

**Power-on sequence**

In general, Flasher STM8 should be powered before connecting the target device. That means you should first connect Flasher STM8 with the host system via USB / ethernet / RS232 and then power up Flasher STM8 if not already powered via USB. Flasher STM8 will boot in "PC" mode. Then connect your target device to Flasher and power up your target device.

**Verifying target device connection with STM8Commander.exe**

If the USB driver is working properly and your Flasher STM8 is connected to the host system, you may connect Flasher STM8 to your target hardware. Then start the STM8 command line tool `STM8Commander.exe`, which should now display the normal Flasher STM8 related information and in addition to that it should report that it found an STM8 target and the target's communication speed. The screenshot below shows the output of `STM8Commander.exe`.

## 4.1.1.2   Setting up Flasher STM8 for stand-alone mode

In order to set up Flasher STM8 for the "stand-alone mode" it has to be in "PC mode". When the correct connection of Flasher STM8 to the host PC is verified start the Flasher STM8 Software. For more information about Flasher STM8 Software, refer to the *Flasher STM8 Software User Guide*. The Flasher STM8 related software and documentation package contains the Flasher STM8 Software. When the Flasher STM8 Software is started, open an appropriate Flasher project file and an appropriate data file for the target you want to program.
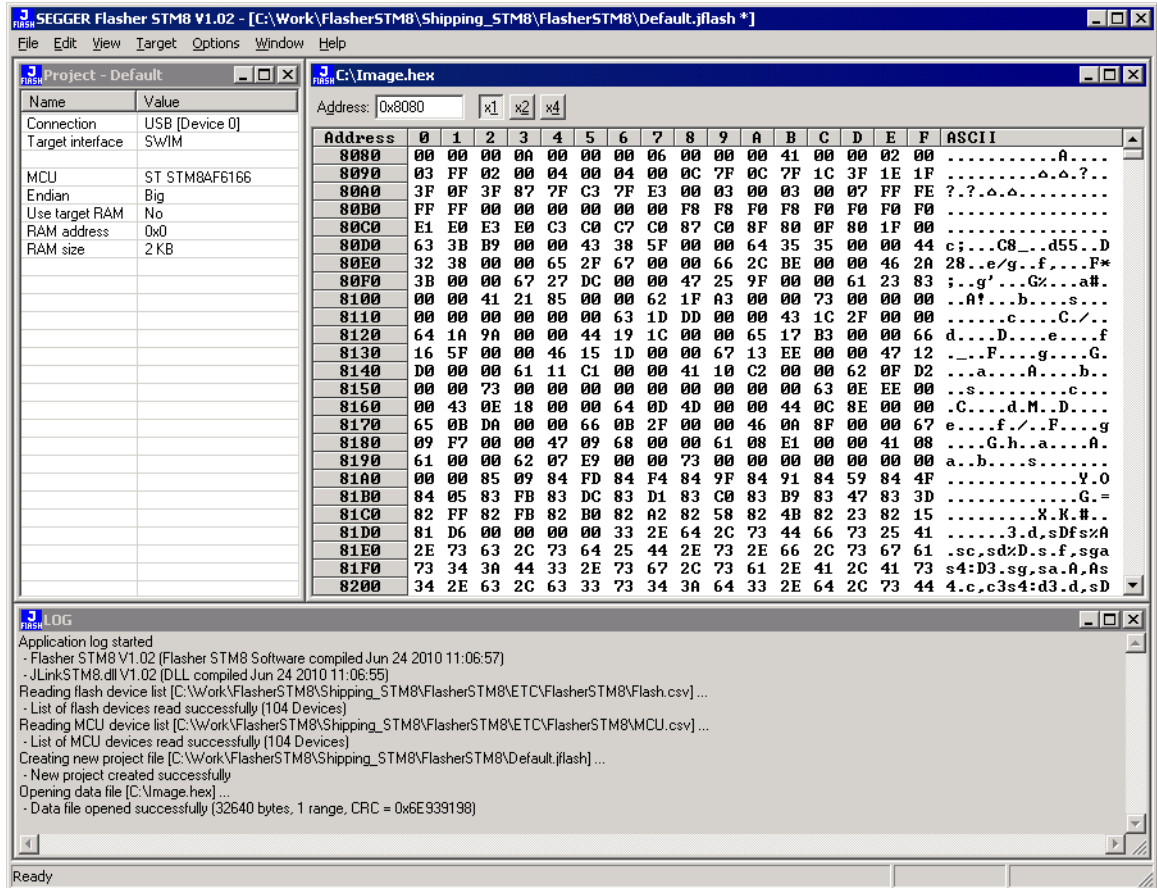
Now, choose **File->Download to programmer** from the menu in order to download the target configuration as well as the data file to the Flasher STM8.
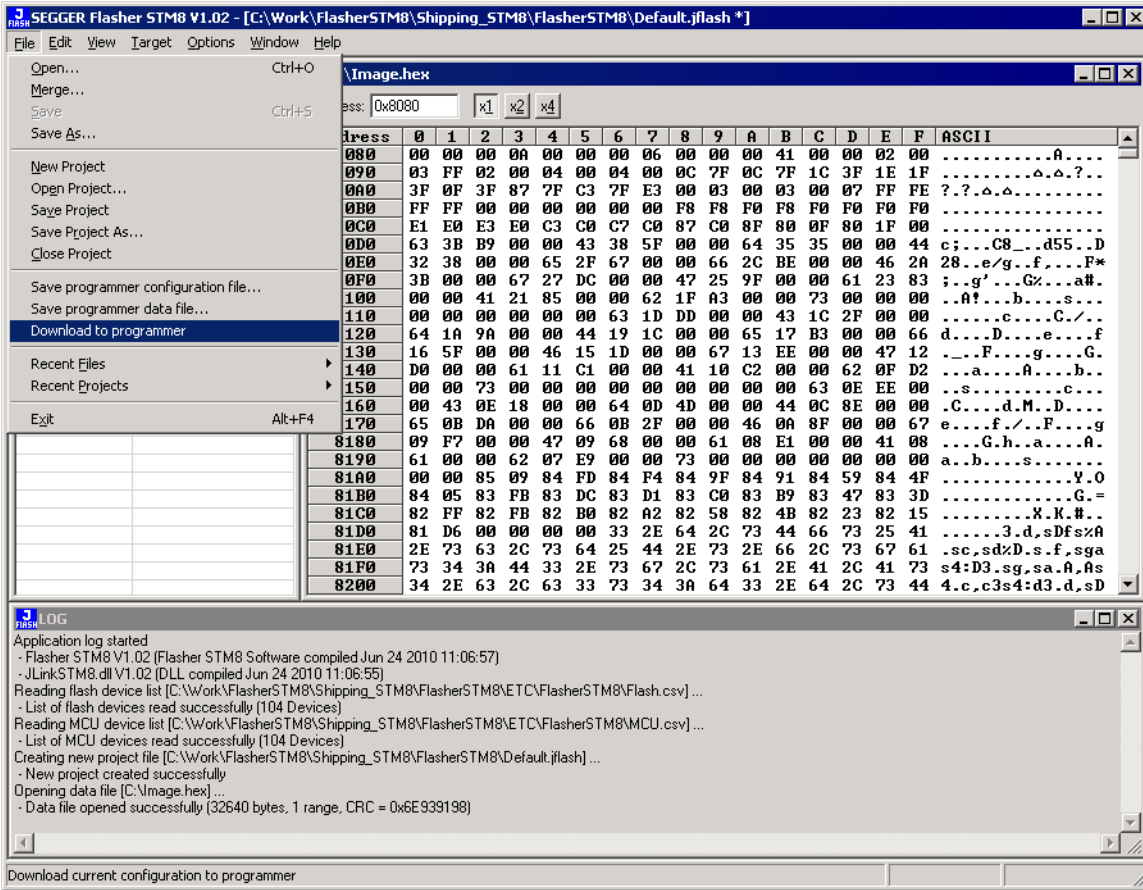
After the download, you should see in the Flasher Log window that the `Flasher.cfg` and the `Flasher.dat` files have been successfully downloaded.
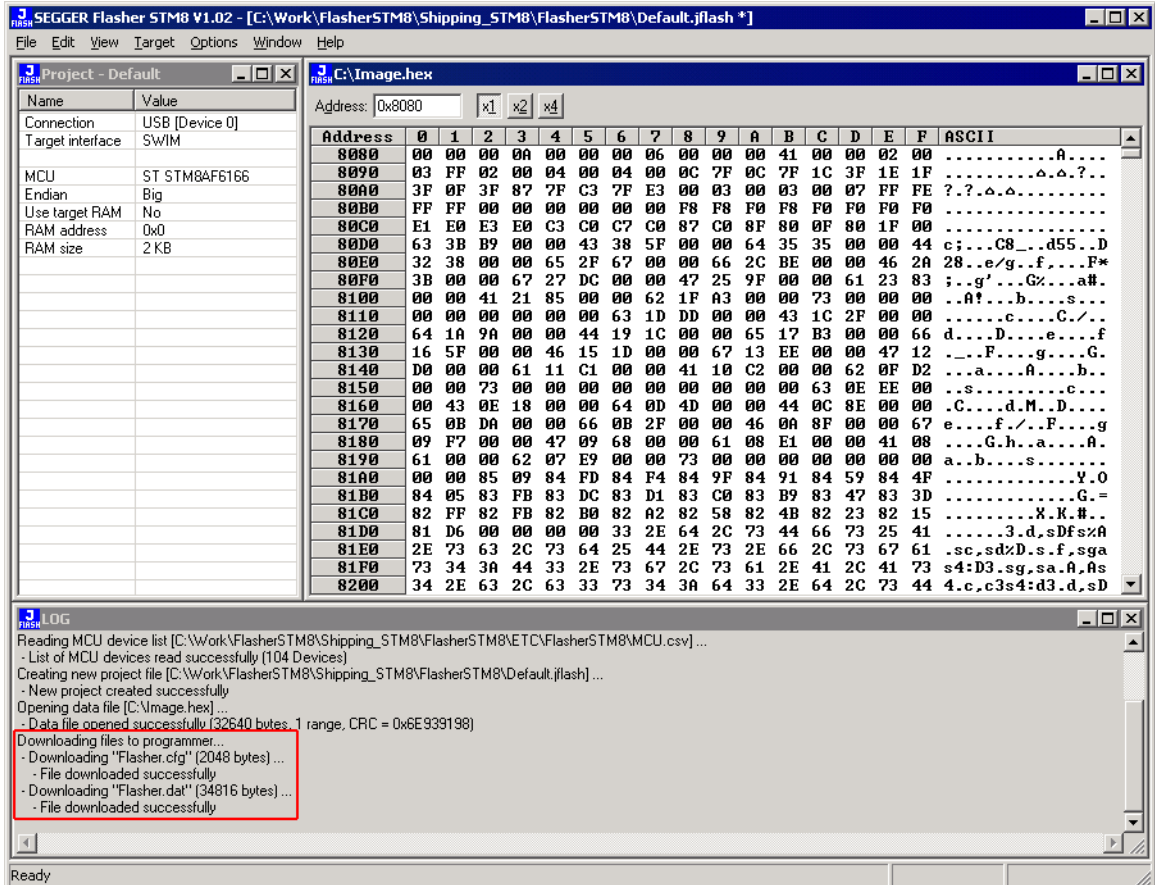


From now on, Flasher STM8 can be used in "stand-alone mode" for stand-alone programming.

## 4.1.2 Stand-alone mode

In order to use Flasher STM8 in "stand-alone mode", it has to be configured first, as described in *Setting up Flasher STM8 for stand-alone mode* on page 35. To boot Flasher STM8 in the "stand-alone mode", just power up the Flasher STM8 without connection to a PC via ethernet or USB. In the "stand-alone mode" Flasher STM8 can be used as a stand-alone flash programmer.

**Note:**    Flasher STM8 can only program the target device it was configured for. In order to program another target device, you have to reconfigure it following the steps described in *Setting up Flasher STM8 for stand-alone mode* on page 35.

### 4.1.2.1 LED status indicators

Progress and result of an operation is indicated by Flasher STM8's LEDs:

| Status of LED | Meaning |
|---|---|
| GREEN, high frequency flashing (10 Hz) | Enumerating Flasher STM8. This only happens before the first programming operation is performed. |
| GREEN, after programming operation has been started | Connect to target and perform init sequence. |

**Table 4.1: Flasher STM8 LEDs**

| Status of LED | Meaning |
|---|---|
| GREEN, slow blinking (1 Hz) | Erasing/Programming/Verifying operation is in progress. |
| GREEN | Operation successful / Ready. |
| RED | Operation failed. |

**Table 4.1: Flasher STM8 LEDs**

## 4.1.3   MSD mode

When pressing the Start/Stop button of Flasher STM8 while connecting it to the PC via USB, Flasher STM8 will boot in the "MSD mode". This mode can be used to down-date a Flasher STM8 firmware version if a firmware update did not work properly and it can be used to configure Flasher STM8 for the "stand-alone mode", without using Flasher STM8 Software. If Flasher STM8 has been configured for "stand-alone mode" before, there will be four files on the MSD, FLASHER.CFG, FLASHER.DAT, FLASHER.LOG, SERIAL.TXT.
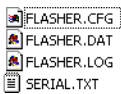


FLASHER.CFG contains the configuration settings for programming the target device and FLASHER.DAT contains the data to be programmed. FLASHER.LOG contains all log-ging information about the commands, performed in stand-alone mode. The SERIAL.TXT contains the serial number, which will be programmed next. Currently, Flasher STM8 Software does not support to configure Flasher STM8 for automated serial number programming.

If you want to configure multiple Flasher STM8 for the same target you do not have to use Flasher STM8 Software all the time. It is also possible to copy the FLASHER.CFG and the FLASHER.DAT files from a configured Flasher STM8 to another one. To copy these files boot Flasher STM8 in "MSD mode".

# 4.2    Multiple File Support

It is also possible to have multiple data files and config files on Flasher STM8, to make Flasher STM8 more easy to use in production environment. To choose the correct configuration file and data file pair, a `FLASHER.INI` file is used. This init file contains a `[FILES]` section which describes which configuration file and which data file should be used for programming. A sample content of a FLASHER.INI file is shown below:

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

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

1.  Boot Flasher STM8 in MSD mode in order to replace the `FLASHER.INI`
2.  If Flasher STM8 is already integrated into the production line, runs in stand-alone mode and can not be booted in other mode: Use the file I/O commands provided by the ASCII interface of Flasher STM8, to replace the `FLASHER.INI`. For more information about the file I/O commands, please refer to *File I/O commands* on page 53.
3.  If Flasher STM8 is already integrated into the production line and is driven via Flasher STM8 PC software: Use the file I/O commands provided by the Flasher STM8 Commander to replace the `FLASHER.INI`. For more information about the STM8 Commander please refer to *STM8 Commander (Command line tool)* on page 29.

# 4.2.1    Setting up configurations and switching configurations

The following steps describe the way to setup and use multiple configurations and how they can be exchanged using the STM8 Commander. Setup and exchanging configurations can be done via RS232 the same way except that the way of file access is slightly different using the ASCII command interface as described in *ASCII command interface* on page 50.

1.  Setup your first configuration you wish to use with the Flasher using Flasher STM8 Software.
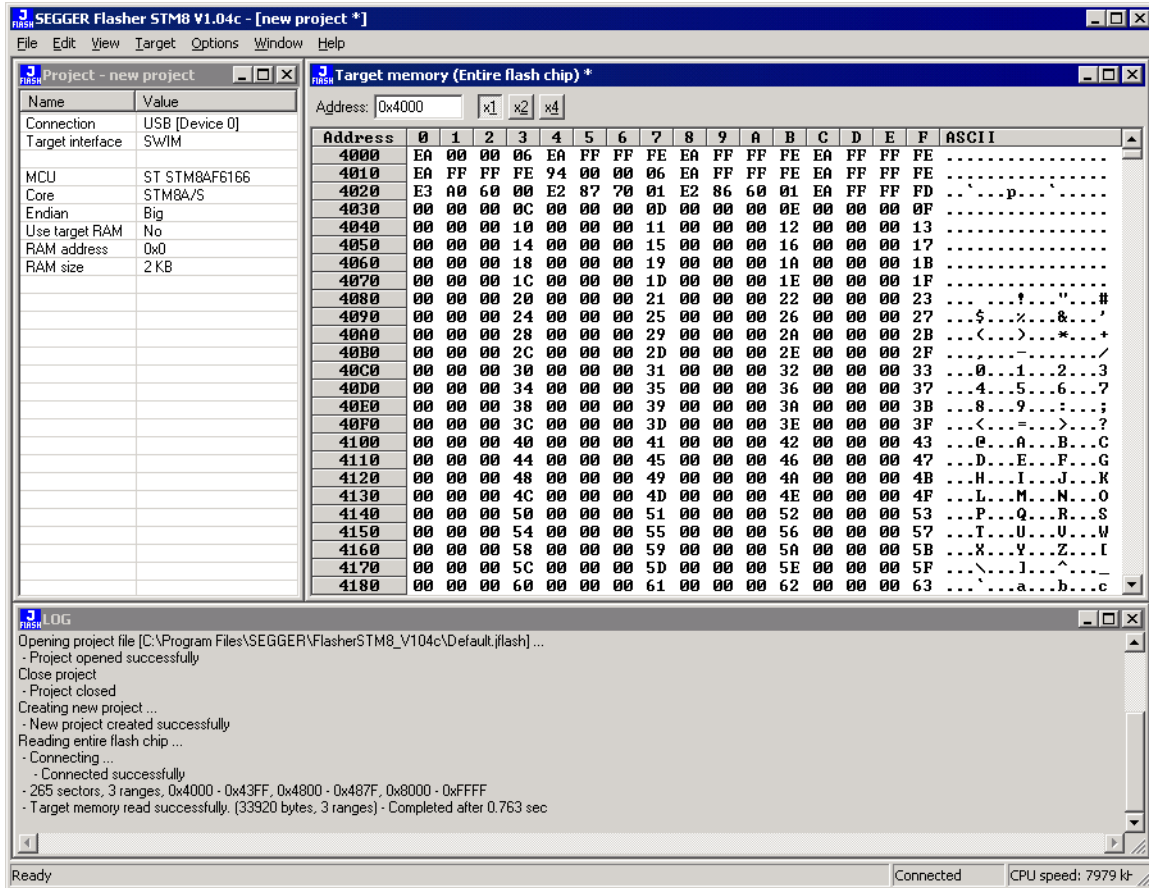
2. Use "File->Save programmer configuration file..." and save as "Flasher1.cfg" to a temporary directory.
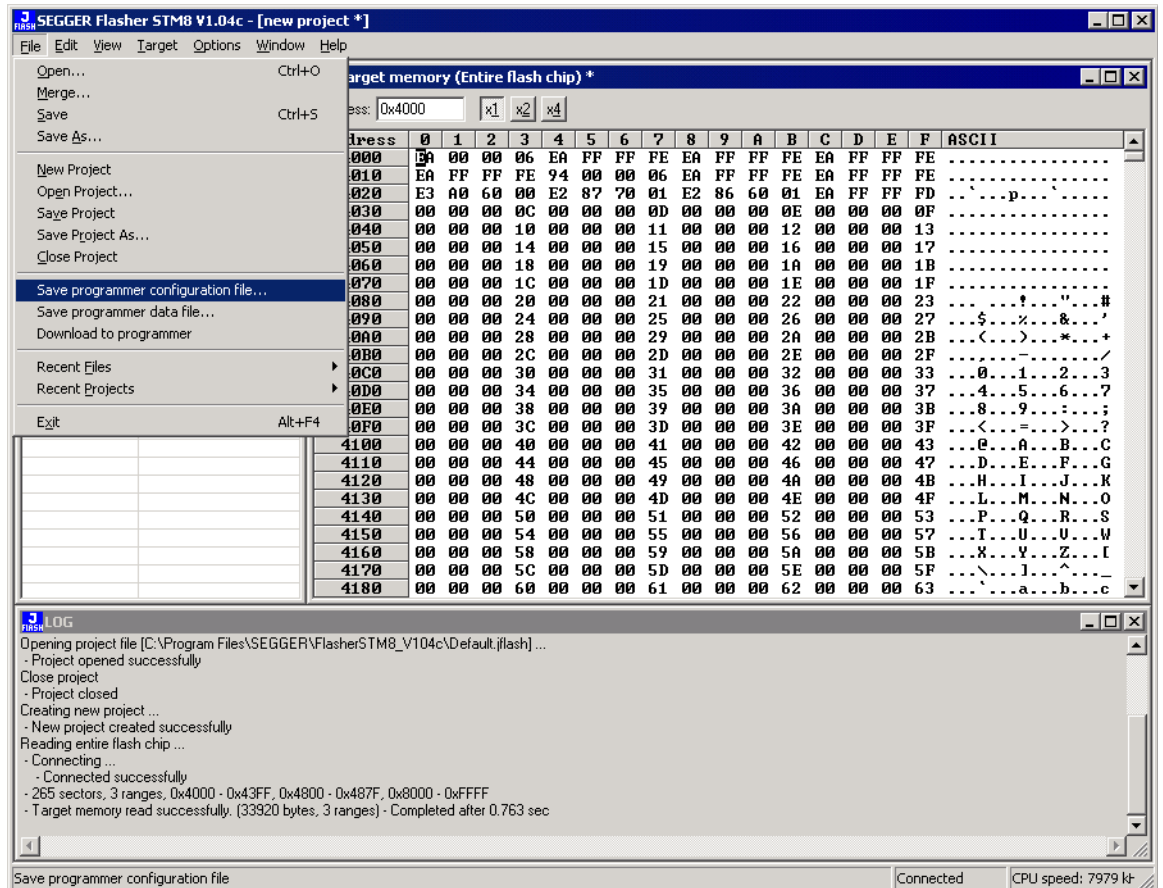


3. Use "File->Save programmer data file..." and save as "Flasher1.dat" to a temporary directory.
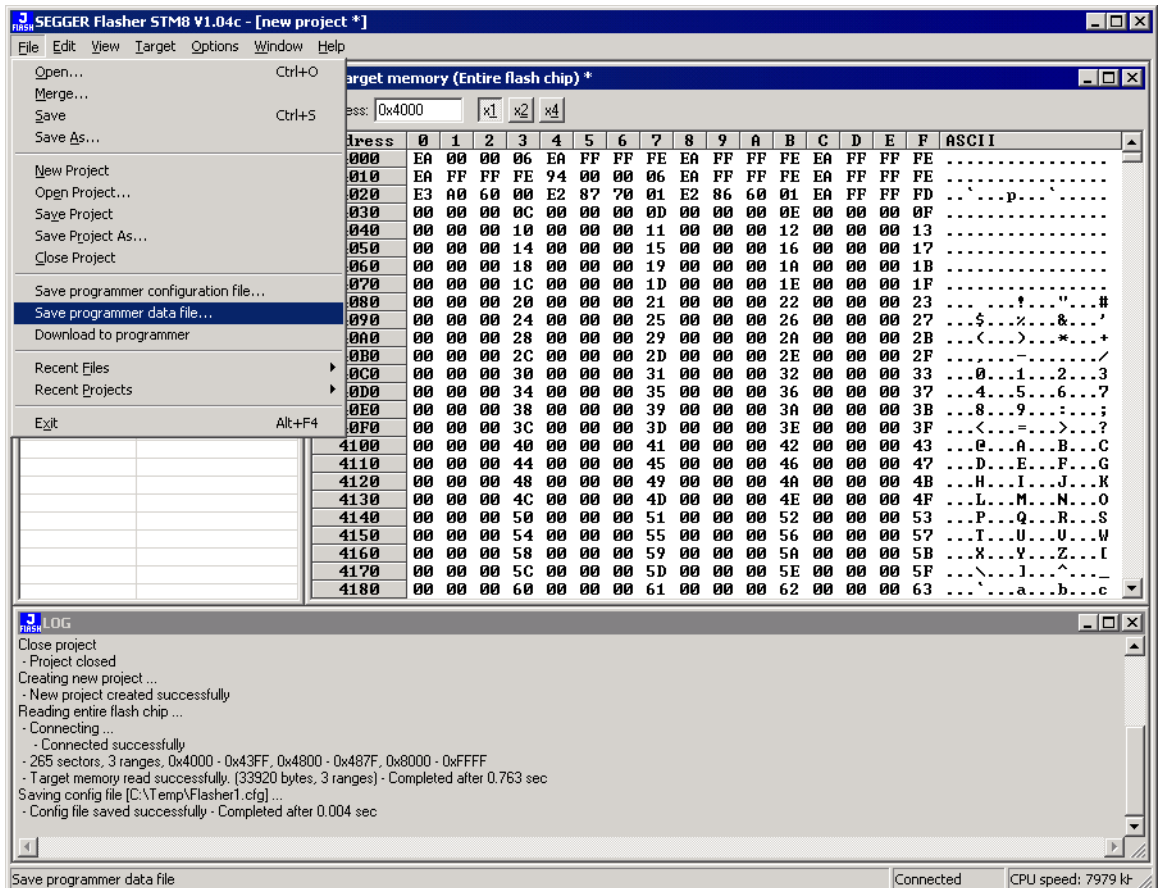
4.  Repeat the steps 1-3 to create an other configuration and save the files as "Flasher2.cfg" and "Flasher2.dat" to the same directory you saved the first set of configuration files.

5.  Create a text file named "FLASHER.ini" with the following content:

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

6.  The content of your temporary folder should now look like the following screenshot:
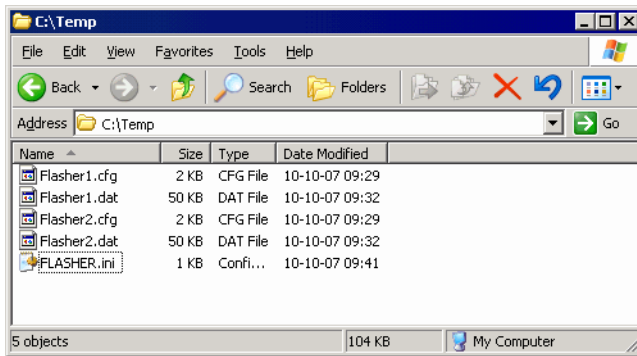


**Note:**      Steps from now on may slightly differ in command usage when using other communication channels such as RS232 ASCII interface but follow the same steps.
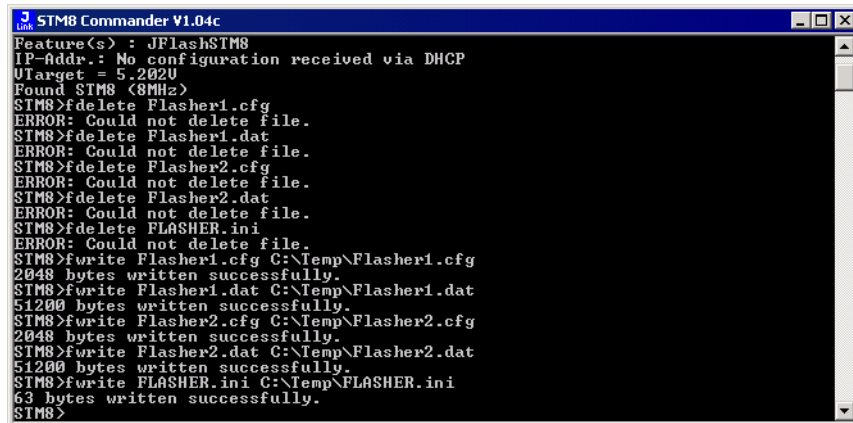
7.  Start the STM8 Commander utility.
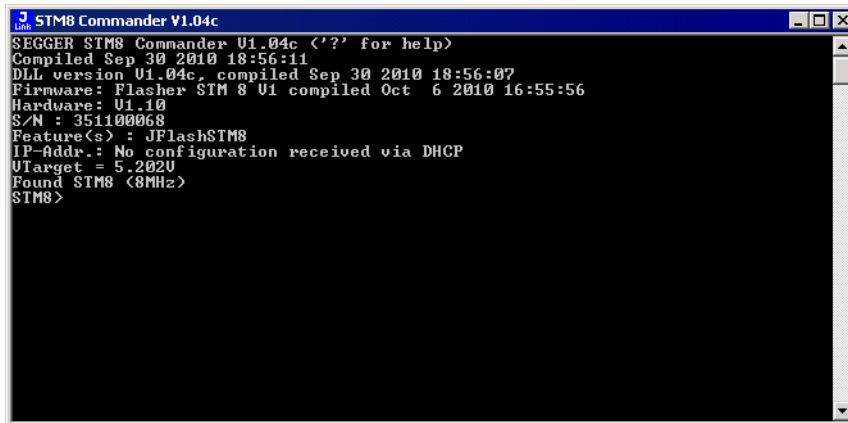


8.  Delete old files on the Flasher using the "fdelete" command if you are unsure if the Flasher is empty. In the screenshot below no files were on the Flasher, therefor returning an error on these operations:
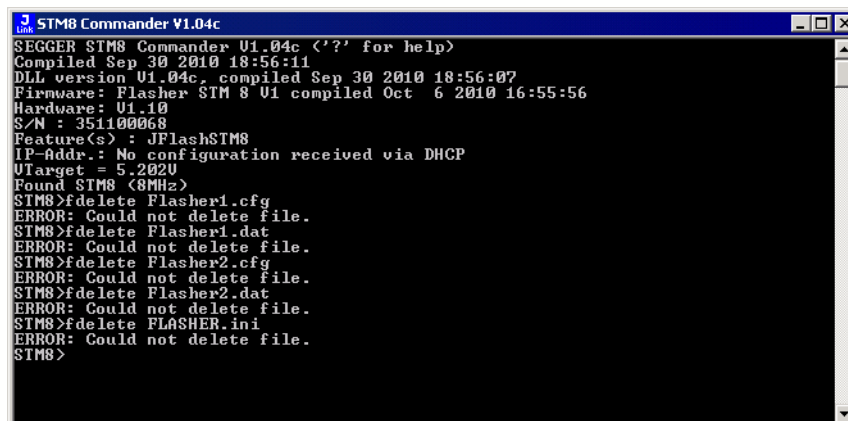
9. Download the files from your temporary folder to the Flasher using the "fwrite" command as shown in the screenshot below:



From now on the files "Flasher1.cfg" and "Flasher1.dat" will be used for stand-alone flashing.

10. If you wish to switch configurations please change your "FLASHER.ini" to contain the file names of the second set of configuration files.
Then delete the old "FLASHER.ini" currently on your Flasher and download the new one to activate the configuration for the second set of files.

# 4.3    Connecting multiple Flasher to your PC

You can connect up to 4 Flasher to your PC. In this case, all Flasher must have different USB-addresses. The default USB-address is 0.

In order to do this, 3 Flasher must be configured as described below. Every Flasher need its own J-Link USB driver which can be downloaded from *www.segger.com*.

**Note:**    Flasher STM8 along with other USB featured SEGGER products use the J-Link USB driver. Therefore you can connect up to 4 devices which use this driver total.

## 4.3.1    How does it work?

USB devices are identified by the OS by their product id, vendor id and serial number. The serial number reported by Flasher is always the same. The product id depends on the configured USB-address.

- The vendor id (VID) representing SEGGER is always 1366
- The product id (PID) for Flasher #1 is 101
- The product id (PID) for Flasher #2 is 102 and so on.

A different PID means that Flasher is identified as a different device, requiring a new driver. The driver for a new Flasher device will be installed automatically.

The sketch below shows a host, running two application programs. Each application communicates with one STM8 core via a separate Flasher.
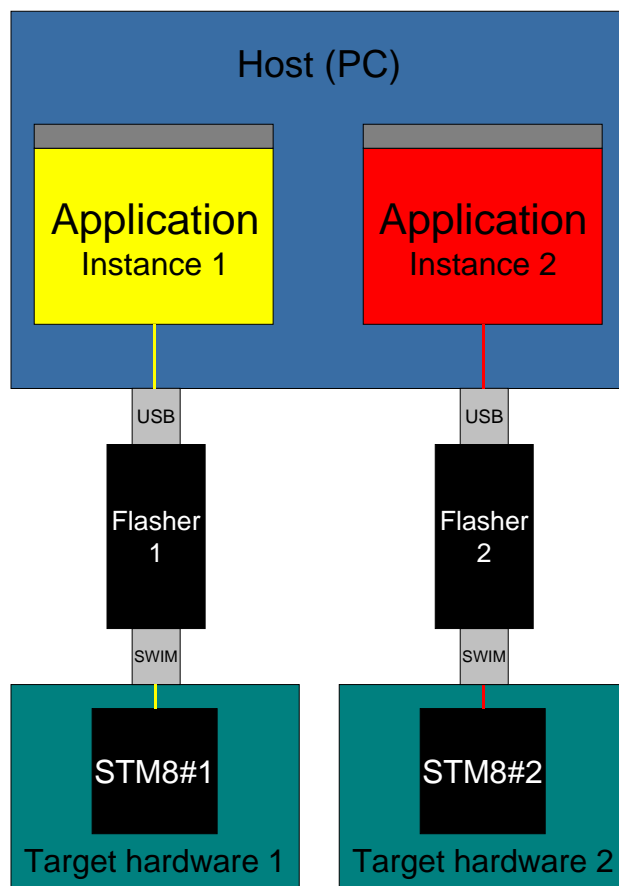
# 4.3.2 Configuring multiple Flasher STM8

1. Start `STM8Commander.exe`
2. Type `usbaddr = 1` to set the Flasher #1.



3. Unplug Flasher and then plug it back in.
4. The system will recognize and automatically install a new Flasher using the J-Link USB driver.
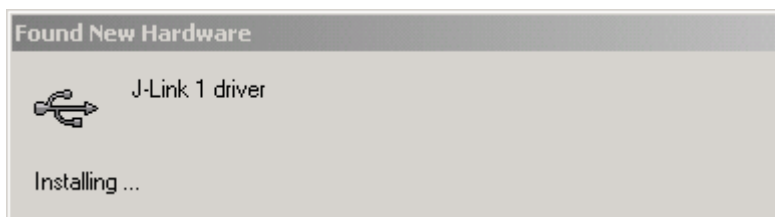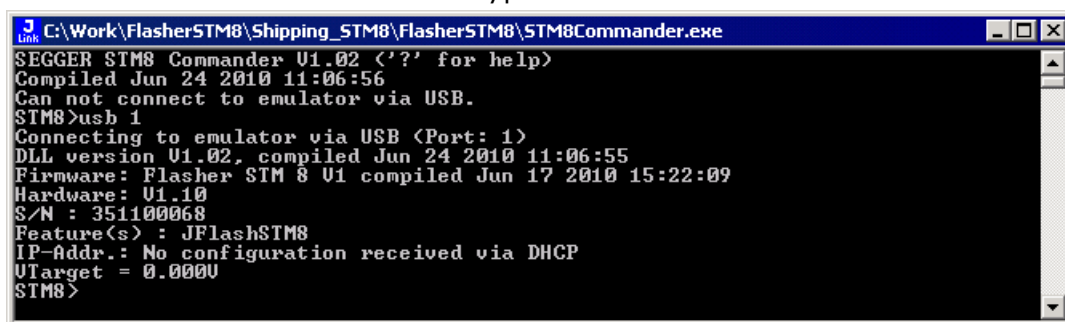


5. You can verify the driver installation by consulting the Windows device manager. If the driver is installed and your Flasher is connected to your computer, the device manager should list the J-Link USB drivers as a node below "Universal Serial Bus controllers" as shown in the following screenshot:
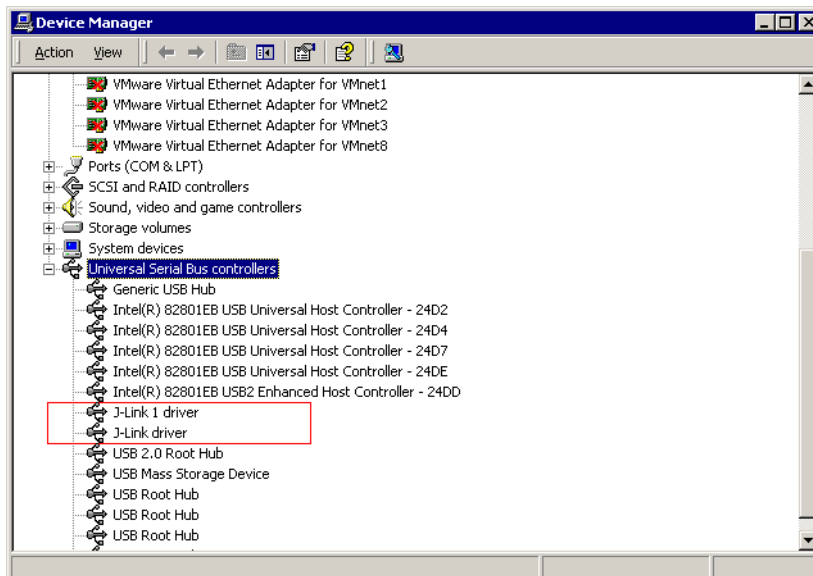
### 4.3.3 Connecting to a Flasher with non default USB-Address

Restart `STM8Commander.exe` and type `usb 1` to connect to Flasher #1.



You may connect other Flasher to your PC and connect to them as well. To connect to an unconfigured Flasher (with default address "0"), restart `STM8Commander.exe` or type `usb 0`.

# Chapter 5

# Remote control

This chapter describes how to control Flasher STM8 via the 9-pin serial interface connector or via TCP/IP.

# 5.1    Overview

There are 4 ways to control Flasher STM8 operation:

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

**Note:**      All four ways to control Flasher STM8 operation can only be used if Flasher STM8 is in standalone mode. In PC / MSD mode they have no effect.

For TCP/IP terminal communication Flasher STM8 switches automatically from PC mode to standalone mode on receiving the first program command via terminal.

# 5.2  Handshake control

Flasher STM8 can be remote controlled by automated testers without the need of a connection to PC and Flasher STM8's PC program. Therefore Flasher STM8 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 STM8:
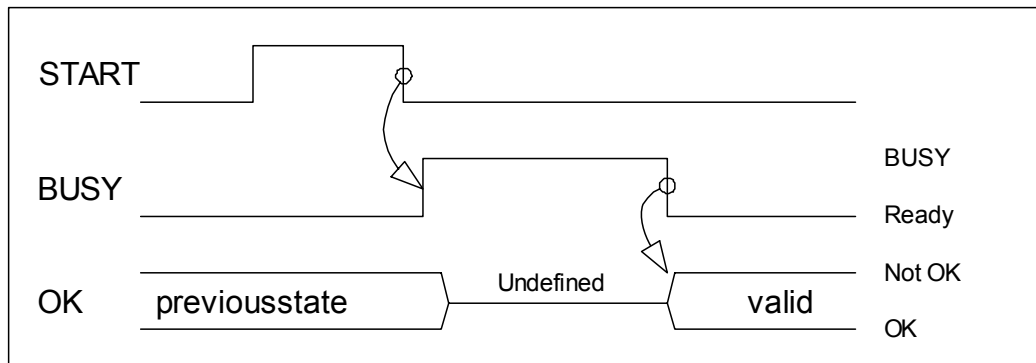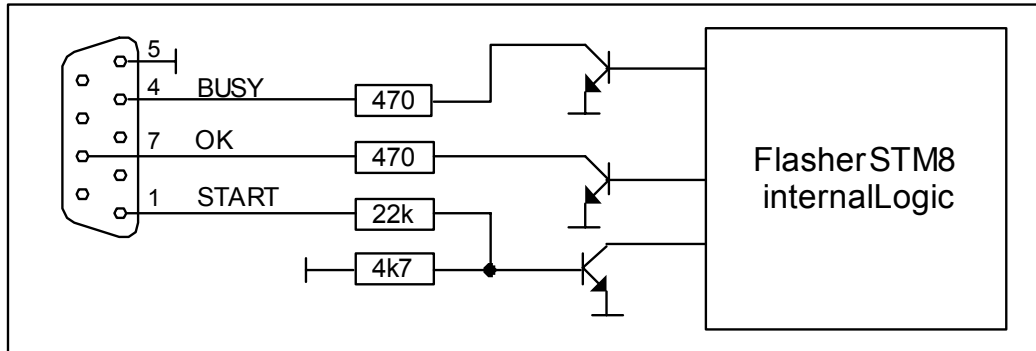




| Pin No. | Function | Description |
|---------|----------|-------------|
| 1 | START | A positive pulse of any voltage between 5 and 30V with duration of min. 30 ms starts "Auto" function (Clear / Program / Verify) on falling edge of pulse. The behavior of the "Auto" function depends on the project settings, chosen in Flasher STM8 Software at the **Production** tab. |
| 4 | BUSY | As soon as the "Auto" function is started, BUSY becomes active, which means that transistor is switched OFF. |
| 5 | GND | Common Signal ground. |
| 7 | OK | This output reflects result of last action. It is valid after BUSY turned back to passive state. The output transistor is switched ON to reflect OK state. |

**Table 5.1: Flasher STM8 LED status**

# 5.3    ASCII command interface

## 5.3.1    Introduction

Once set up using Flasher STM8 Software, Flasher STM8 can be driven by any application or just a simple terminal using ASCII commands.

Every known command is acknowledged by Flasher and then executed. After command execution, Flasher sends an ASCII reply message. If an unknown command is received, Flasher responds with #NACK.

## 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 (ASCII code 13)
*   Commands can be sent upper or lower case.

## 5.3.3    Communication port settings

Flasher can be driven via TCP/IP or via a RS232 serial port with the following interface settings:

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

at 9600 baud.

## 5.3.4    Commands to Flasher

The following commands are supported by the current version of Flasher firmware:

**#AUTO**

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

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

*   Flasher starts erasing
*   Flasher programs target CPU
*   Flasher verifies target CPU

Depending on the settings chosen in the **Production** tab in Flasher STM8 Software, this sequence can differ from the one shown above.

Finally, Flasher responds with

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

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

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

### #AUTO 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`

### #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)
```

### #START

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

Flasher will reply with the following sequence of messages:

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

### #STATUS

This command can be sent any time, even during other command execution. Flasher responds with its current state. All defined state messages are described under *Reply from Flasher STM8* on page 56.

### #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.

### #VERIFY

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

### #RESULT

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

### #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.
```

### #BAUDRATE<Baudrate>

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

If this command succeeds, Flasher responds with:

```
#ACK
#OK
```

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

```
#ERR255: Invalid parameters
```

or

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

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

### #SELECT <Filename>

This command can be sent in order to change the content of the `FLASHER.INI` file to select an other set of data and configuration file already present on the Flasher. For further information regarding working with multiple files please refer to *Multiple File Support* on page 39. The <Filename> parameter is the name of the *.dat and *.cfg file to be used without their extension.

```
#SELECT FILE1
```

This command rewrites the `FLASHER.INI` to use the following settings:

```
[FILES]
DataFile = "FILE1.DAT"
ConfigFile = "FILE1.CFG"
```

Flasher responds with:

```
#ACK
#OK
```

### #POWER <ON/OFF> [PERM]

This command can be sent in order to activate/deactivate the 5V power supply from Flasher to target. The optional `PERM` parameter writes this selection into the Flasher as default when powering the Flasher.

A typical sequence when using the `#POWER` command looks as follows:

```
#POWER ON
#ACK
#OK
```

### 5.3.4.1 File I/O commands

The ASCII interface of Flasher STM8 also supports file I/O operations via terinal. The following file I/O commands are supported:

#### #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 sent and another file is already open, Flasher will respond with:

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

#### #FCLOSE

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

A typical sequence when using the `#FCLOSE` command looks 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 error reply:

```
#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 the Flasher STM8 Software when downloading or saving configuration or data files.

A typical sequence when using the `#FCRC` command looks 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 looks 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
```

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

The #FWRITE command is used to write a file. <Offset> specifies the offset in the file, at which data writing is started. <NumBytes> specifies the number of bytes which are sent 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 sent: 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
```

## #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 looks as follows:

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

If the #FREAD command succeeds, Flasher will finally respond with a #OK:<Num-Bytes>:<Data> reply message. For more information about the Flasher reply messages, refer to *Reply from Flasher STM8* on page 56.

**Note:**      In order to use the #FREAD command, a file has to be opened before, using 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 looks as follows:

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

If the #FSIZE command succeeds, Flasher will respond with a #OK:<Size> reply message. For more information about the Flasher reply messages, please refer to *Reply from Flasher STM8* on page 56.

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

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

## #FLIST

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

A typical sequence when using the `#FLIST` command looks 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
```

## #FORMATHIGH

The `#FORMATHIGH` command is used to format the internal storage of the Flasher.

A typical sequence when using the `#FORMATHIGH` command looks as follows:

```
#FORMATHIGH
#ACK
#OK
```

## #FORMATLOW

The `#FORMATLOW` command is used to low-level format the internal storage of the Flasher. This can be used in case the internal storeage gets corrupted in a way that it can not simple be high-level formatted. After low-level format a high-level format needs to be done.

A typical sequence when using the `#FORMATLOW` command looks as follows:

```
#FORMATLOW
#ACK
#OK
```

## 5.3.4.2 Target commands

The ASCII interface of Flasher STM8 also supports direct target operations via terminal. The following file I/O commands are supported:

### #READMEM <Address>,<NumBytes>

The #READMEM command is used to read memory directly from the STM8. <Address> specifies the start address at which data reading is started. <NumBytes> specifies the number of bytes which should be read.

A typical sequence when using the #READMEM command looks as follows:

```
#READMEM 4800,3
#ACK
#OK:03:0000FF
```

If the #READMEM command succeeds, Flasher will finally respond with a #OK:<Num-Bytes>:<Data> reply message. For more information about the Flasher reply messages, refer to *Reply from Flasher STM8* on page 56.

# 5.3.5 Reply from Flasher STM8

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

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

The following reply messages from Flasher are defined:

### #ACK

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

### #NACK

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

### #OK

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

### #STATUS:

Flasher replies with its current state.

The following status messages are currently defined:

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

**Table 5.2: List of status messages that are currently defined**

**#ERRxxx**

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

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

For example:

`#ERR007:CANCELED.`

The error code numbers are described in the following table:

| Message | Description |
|---------|-------------|
| `#ERR007` | Flasher received `#CANCEL` command and has canceled the current operation. |
| `#ERR255` | Undefined error occurred. This reply is followed by an error string. |

**Table 5.3: List of error code numbers which are currently defined**
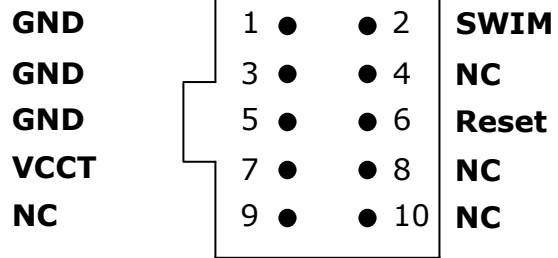
# Chapter 6

# Hardware

This chapter gives an overview about Flasher STM8 specific hardware details, such as the pinouts and available adapters.

# 6.1    10-pin connector

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

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

| | | |
|---|---|---|
| **GND** | 1 ● | ● 2 | **SWIM** |
| **GND** | 3 ● | ● 4 | **NC** |
| **GND** | 5 ● | ● 6 | **Reset** |
| **VCCT** | 7 ● | ● 8 | **NC** |
| **NC** | 9 ● | ● 10 | **NC** |

## 6.1.1    10-pin connector pinout

The following table lists the Flasher STM8 10-pin connector pinout.

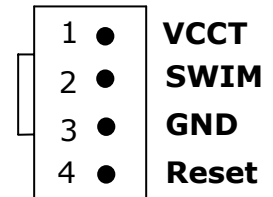| PIN | SIGNAL | TYPE | Description |
|---|---|---|---|
| 2 | SWIM | I/O | Open drain output, tristateable, with 100 Ohms series resistor to target.<br>This is the pin used to transfer SWIM input/output data.<br>In case of connection problems the SWIM pin should be pulled up with a 470-680 Ohms series resistor as it can be found in several reference designs. |
| 6 | Reset | I/O | Open collector with 100 Ohms series resistor to target.<br>Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called "nRST", "nRESET" or "RESET". |
| 7 | VCCT | I/O | Target voltage reference or target supply pin.<br>Has to be connected to the target CPUs supply voltage. It may be used to supply the target board. |

**Table 6.1: Flasher STM8 10-pin connector pinout**

Pins 1, 3, 5 are GND pins connected to GND in Flasher STM8. They should also be connected to GND in the target system.

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

## 6.2 4-pin Connector

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

```
1  ●  │  VCCT
2  ●  │  SWIM
3  ●  │  GND
4  ●  │  Reset
```

### 6.2.1 4-pin connector pinout

The following table lists the Flasher STM8 4-pin connector pinout.

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

**Table 6.2: Flasher STM8 4-pin connector pinout**

Pins 3 is GND pin connected to GND in Flasher STM8. It should also be connected to GND in the target system.

# 6.3 Connection cable

The standard cable which is delivered with Flasher STM8 to connect a target to the Flasher is a 4 way ribbon cable of type ERNI part no. 839016 (100mm) mounted with keyed header (female) mounted on both sides (resulting in a total length of around 120mm).

Flasher STM8 is tested with the delivered cable for being fully functional. Using your own cable or using a different cable length is not recommended as proper function can not be guaranteed. Using a different cable is on your own risk.

# 6.4 Target board design for SWIM

We strongly advise following the recommendations given by the chip manufacturer. These recommendations are normally in line with the recommendations given in the table *10-pin connector pinout* on page 60 and *4-pin connector pinout* on page 61. In case of doubt you should follow the recommendations given by the semiconductor manufacturer.

To be on the safe side SEGGER recommends the GND connections of the Flasher to be always connected to the the ground of the power supply of the target to suppress spikes during connecting or disconnecting your target hardware to/from Flasher.

**Note:** Flasher STM8 uses 100 Ohms resistor on SWIM and reset line. We expect the target hardware to use pull-up resistors between 2.2 kOhms to 10 kOhms on both lines.

## 6.4.1 Target power supply

Pin 1 of the connector can be used to supply power to the target hardware. Supply voltage is 5V, max. current is 300mA. The output current is monitored and protected against overload and short-circuit.

Power can be controlled via commands using the STM8 Commander. The following commands are available to control power:

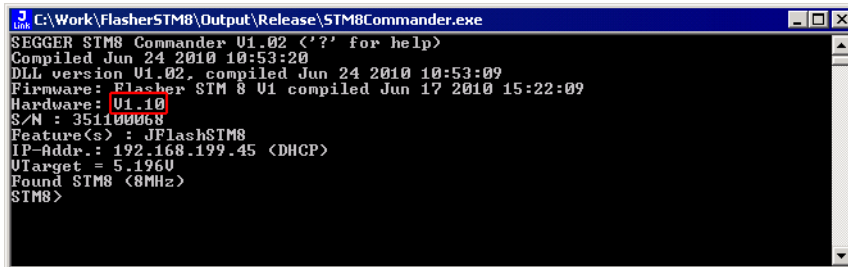| Command | Explanation |
|---|---|
| power on | Switch target power on |
| power off | Switch target power off |
| power on perm | Set target power supply default to "on" |
| power off perm | Set target power supply default to "off" |

**Table 6.3: Command List**

# 6.5    How to determine the hardware version

To determine the hardware version of your Flasher STM8, you should first look at the label at the bottom side of the unit. The hardware version is printed on the back label.

If this is not the case with your Flasher STM8, you can use `STM8Commander.exe` to determine your hardware version (if Flasher STM8 is in PC mode). As part of the initial message, the hardware version is displayed. For more information about how to ensure that Flasher STM8 is in PC mode, please refer to *PC mode* on page 34.

# Chapter 7

# Background information

This chapter provides background information about flash programming in general. It also provides information about how to replace the firmware of Flasher STM8 manually.

# 7.1    Flash programming

Flasher STM8 comes with a DLL, which allows - amongst other functionalities - reading and writing RAM and starting and stopping the CPU.

## 7.1.1    How does flash programming via Flasher STM8 work ?

This requires extra code. This extra code typically downloads a program into the RAM of the target system, which is able to erase and program the flash. This program is called RAM code and "knows" how to program the flash; it contains an implementation of the flash programming algorithm for the particular device. The RAM code requires data to be programmed into the flash memory. The data is supplied by downloading it to RAM.

An other way to program is in circuit programming. This is done by writing to the specific registers that need to be programmed via SWIM directly. Depending on the flash type and size of the flash this may result in slower programming speed then programming with usage of RAM (RAM code usage).

## 7.1.2    Data download to RAM

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

## 7.1.3    Available options for flash programming

In general, there are two possibilities in order to use Flasher STM8 for flash programming:

*   Using Flasher STM8 stand-alone to program the target flash memory (stand-alone mode)
*   Using Flasher STM8 in combination with Flasher STM8 Software to program the target flash memory (Flasher STM8 in "PC mode")

### 7.1.3.1    Using Flasher STM8 in stand-alone mode

In order to use the Flasher STM8 in stand-alone mode, it has to be configured first. For more information about how to setup Flasher STM8 for using in "stand-alone mode", please refer to *Setting up Flasher STM8 for stand-alone mode* on page 35.

### 7.1.3.2    Flasher STM8 Software - Complete flash programming solution

Flasher STM8 Software is a stand-alone Windows application, which can read / write data files and program the flash in all STM8 systems. For more information about Flasher STM8 Software please refer to the *Flasher STM8 Software User Guide*, which can be downloaded from our website *http://www.segger.com*.

# Chapter 8

# Support and FAQs

This chapter contains troubleshooting tips together with solutions for common problems which might occur when using Flasher STM8. 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 STM8 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 STM8.

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

- A detailed description of the problem
- Flasher STM8 serial number
- Information about your target hardware (processor, board, etc.).
- `FLASHER.CFG`, `FLASHER.DAT`, `FLASHER.LOG`, `SERIAL.TXT` file from Flasher STM8. To get these files, Flasher STM8 has to be in MSD mode. For more information about how to boot Flasher STM8 in MSD mode, please refer to *MSD mode* on page 38.

Flasher STM8 is sold directly by SEGGER.

## 8.2    Frequently Asked Questions

**Maximum download speed**

Q:    What is the maximum download speed supported by Flasher STM8 ?

A:    Flasher STM8's maximum supported programming speed depends on flash pro-
gramming speed supported by your target chip.
Programming a 128kByte device can be done in less than 9 seconds resulting in
speeds of about 14-15 kByte/sec.

# 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.

### Coprocessor

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

### Host

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

### ID

Identifier.

### Image

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

### 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.

### Target

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

# Chapter 10

# Literature and references

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

| Reference | Title | Comments |
|-----------|-------|----------|
| [Flasher STM8 Software] | Flasher STM8 Software User Guide | This document describes the Flasher STM8 Software. It is publicly available from SEGGER (*www.segger.com*). |

**Table 10.1: Literature and References**

# Index