Flasher RX

User guide of the stand-alone JTAG programmer for Renesas RX Cores

Manual Rev. 0

Date: February 7, 2011

Document: UM08015

SEGGER

A product of SEGGER Microcontroller GmbH & Co. KG

www.segger.com

Disclaimer

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

Copyright notice

Trademarks

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

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

Contact address

SEGGER Microcontroller GmbH & Co. KG

In den Weiden 11
D-40721 Hilden

Germany

Tel.+49 2103-2878-0
Fax.+49 2103-2878-28
Email: support@segger.com
Internet: *http://www.segger.com*

Revisions

This manual describes the Flasher RX device.

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

Revision	Date	By	Explanation
0	110207	AG	Initial version.

About this document

This document describes the Flasher RX. It provides an overview over the major features of the Flasher RX, gives you some background information about JTAG and describes Flasher RX related software packages available from Segger. Finally, the chapter *Support and FAQs* on page 43 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 & Co. KG develops and distributes software development tools and ANSI C software components (middleware) for embedded systems in several industries such as telecom, medical technology, consumer electronics, automotive industry and industrial automation.

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

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

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

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

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

EMBEDDED SOFTWARE (Middleware)

emWin
Graphics software and GUI

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

embOS
Real Time Operating System

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

emFile
File system

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

emUSB
USB device stack

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

SEGGER TOOLS

Flasher
Flash programmer

Flash Programming tool primarily for microcontrollers.

J-Link
JTAG emulator for ARM cores

USB driven JTAG interface for ARM cores.

J-Trace
JTAG emulator with trace

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

J-Link / J-Trace Related Software

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

Table of Contents

Chapter 1

Introduction

This chapter gives a short overview about the Flasher RX.

1.1 Flasher RX overview

Flasher RX is a programming tool for microcontrollers with on-chip or external Flash memory and Renesas RX610, RX621, RX62N, RX62T core. Flasher RX is designed for programming flash targets with the J-Flash software or stand-alone. In addition to that Flasher RX can also be used as a J-Link. For more information about J-Link please refer to the *J-Link / J-Trace User Guide* which can be downloaded at *http://www.segger.com*.
Flasher RX connects to a PC using the USB/Ethernet/RS232 interface, running Microsoft Windows 2000, Windows XP, Windows 2003, Windows Vista or Windows 7. In stand-alone mode, Flasher can be driven by the start/stop button, or via the RS232 interface (handshake control or ASCII interface). Flasher RX itself has a 20-pin JTAG connector but comes with a 14-pin adapter for Renesas RX.

1.1.1 Features of Flasher RX

- Three boot modes: J-Link mode, stand-alone mode, MSD mode
- Stand-alone JTAG programmer (Once set up, Flasher can be controlled without the use of PC program)
- No power supply required, powered through USB
- Support for Renesas RX610, RX621, RX62N, RX62T core
- Supports internal and external flash* devices
- 64 MB memory for storage of target program
- Can be used as J-Link (JTAG emulator) with a download speed of up to 720 Kbytes/second
- Programming speed between 170 and 300 Kbytes/second depending on target hardware
- Serial in target programming supported
- Data files can updated via USB/Ethernet (using the J-Flash software), via RS232 or via the MSD functionality of Flasher RX
- Target interface: JTAG

*Comming soon

1.1.2 Working environment

General

Flasher RX 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 182MB of RAM, running Microsoft Windows 2000, Windows XP, Windows 2003, Windows Vista or Windows 7. It needs to have a USB or RS232 interface available for communication with Flasher RX.

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 RX PC-software J-Flash

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

1.2 Specifications

1.2.1 Specifications for Flasher RX

General	
Supported OS	Microsoft Windows 2000 Microsoft Windows XP Microsoft Windows XP x64 Microsoft Windows 2003 Microsoft Windows 2003 x64 Microsoft Windows Vista Microsoft Windows Vista x64 Microsoft Windows 7 Microsoft Windows 7 x64
Operating Temperature	+5 °C ... +60 °C
Storage Temperature	-20 °C ... +60 °C
Relative Humidity (non-condensing)	<90% rH
Mechanical	
Size (without cables)	121mm x 66mm x 30mm
Weight (without cables)	119g
Available interfaces	
USB Host interface	USB 2.0, full speed
Ethernet Host interface	10/100 MBit
RS232 Host interface	RS232 9-pin
Target interface	JTAG 20-pin (shipped with 14-pin adapter for Renesas RX)
JTAG Interface, Electrical	
Power Supply	USB powered, 100mA for Flasher ARM. 500 mA if target is powered by Flasher ARM
Target interface voltage (VIF)	1.2 ... 5V
Target supply voltage	4.5V...5V (on the 14-pin adapter the target supply voltage can be switched between 3.3V and 5V)
Target supply current	max. 400mA
Reset Type	Open drain. Can be pulled low or tristated
Reset low level output voltage (V_{OL})	V_{OL} <= 10% of V_{IF}
For the whole target voltage range (1.8V <= V_{IF} <= 5V)	
LOW level input voltage (V_{IL})	V_{IL} <= 40% of V_{IF}
HIGH level input voltage (V_{IH})	V_{IH} >= 60% of V_{IF}
For 1.8V <= V_{IF} <= 3.6V	
LOW level output voltage (V_{OL}) with a load of 10 kOhm	V_{OL} <= 10% of V_{IF}
HIGH level output voltage (V_{OH}) with a load of 10 kOhm	V_{OH} >= 90% of V_{IF}
For 3.6 <= V_{IF} <= 5V	
LOW level output voltage (V_{OL}) with a load of 10 kOhm	V_{OL} <= 20% of V_{IF}

Table 1.1: Flasher RX specifications

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

Table 1.1: Flasher RX specifications

1.2.2 Flasher RX Download speed

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

Hardware	RX600 series *Memory* download
Flasher RX Rev. 1	720 Kbytes/s (12MHz JTAG)

Table 1.2: Download speed differences between hardware revisions

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

Chapter 2

Working with Flasher RX

This chapter describes functionality and how to use Flasher RX.

2.1 Operating modes

Flasher RX is able to boot in 3 different modes:

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

If Flasher RX can establish an Ethernet uplink or can enumerate on the USB port, it boots in "J-Link mode". In this mode, Flasher RX can be used as a J-Link. When Flasher RX is but can not establish a connection with the host, the "stand-alone mode" is started. In this mode Flasher RX can be used as a stand-alone flash programmer. When the Start/Stop button is kept pressed when power supply is enabled, Flasher RX boots in "MSD mode". In this mode, Flasher RX boots as a mass storage device.

2.1.1 J-Link mode

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

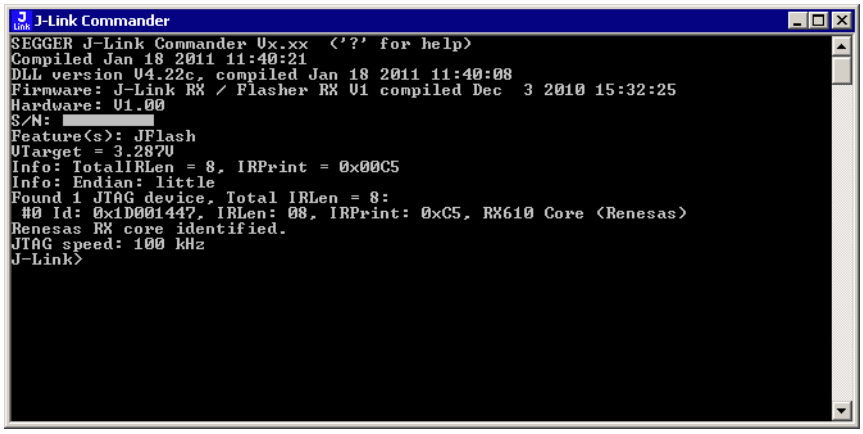
2.1.1.1 Connecting the target system

Power-on sequence

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

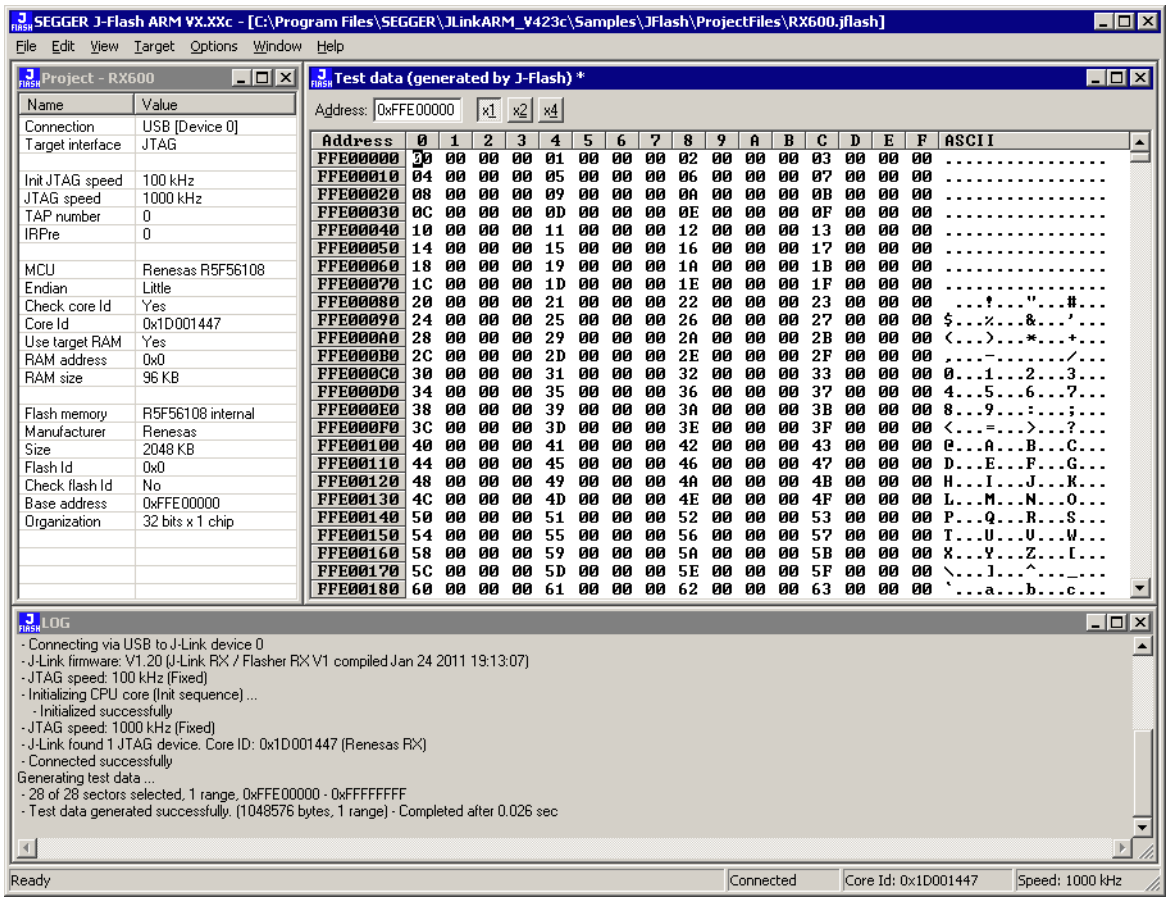
Verifying target device connection with J-Link.exe

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

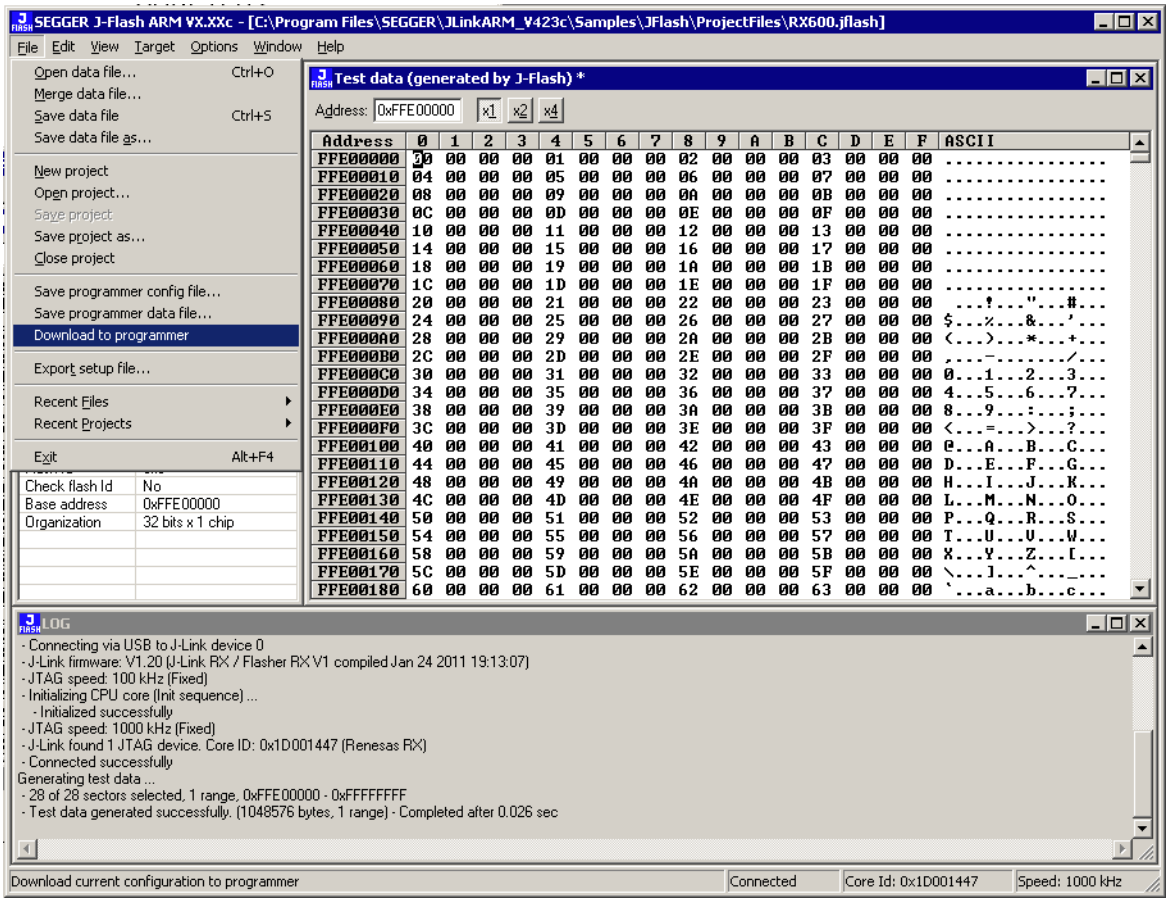


2.1.1.2 Setting up Flasher RX for stand-alone mode

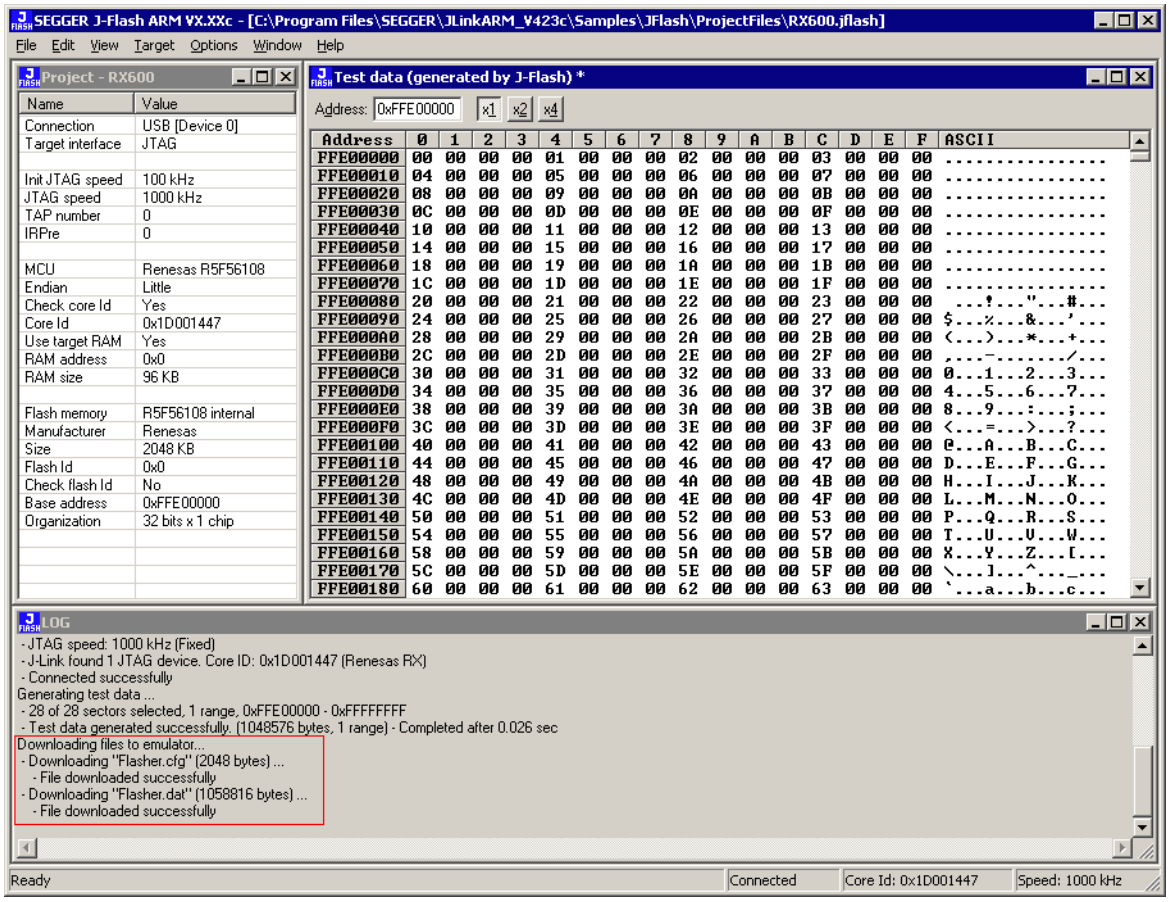
In order to set up Flasher RX for the "stand-alone mode" it has to be in "J-Link mode". When the correct connection of Flasher RX to the host PC is verified start the J-Flash software. For more information about J-Flash, please refer to the *J-Flash User Guide*. When J-Flash is started, open an appropriate J-Flash project file and an appropriate data file for the target you want to program with Flasher RX.



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 RX.



After the download, you should see in the J-Flash Log window that the `Flasher.cfg` and the `Flasher.dat` files have been successfully downloaded.



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

2.1.2 Stand-alone mode

In order to use Flasher RX in "stand-alone mode", it has to be configured first, as described in *Setting up Flasher RX for stand-alone mode* on page 13. To boot Flasher RX in the "stand-alone mode", only the power supply to Flasher RX has to be enabled (Flasher RX should not be connected to a PC). In the "stand-alone mode" Flasher RX can be used as a stand-alone flash programmer.

Note: Flasher RX can only program the target device it was configured for. In order to program another target device, you have to repeat the steps described in *Setting up Flasher RX for stand-alone mode* on page 13.

2.1.2.1 LED status indicators

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

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

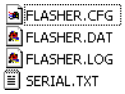
Table 2.1: Flasher RX LEDs

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

Table 2.1: Flasher RX LEDs

2.1.3 MSD mode

When pressing the Start/Stop button of Flasher RX while connecting it to the PC, Flasher RX will boot in the "MSD mode". This mode can be used to downdate a Flasher RX firmware version if a firmware update did not work properly and it can be used to configure Flasher RX for the "stand-alone mode", without using J-Flash. If Flasher RX 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 logging information about the commands, performed in stand-alone mode. The `SERIAL.TXT` contains the serial number, which will be programmed next. Currently, J-Flash does not support to configure Flasher RX for automated serial number programming.

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

2.2 Multiple File Support

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

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

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

1. Boot Flasher RX in MSD mode in order to replace the `FLASHER.INI`
2. If Flasher RX 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 RX, to replace the `FLASHER.INI`. For more information about the file I/O commands, please refer to *File I/O commands* on page 28.

2.3 Target interfaces

Currently the following target interfaces are supported by Flasher RX:

- JTAG

For more information about the target interfaces itself, please refer to *UM08001, chapter "Working with J-Link and J-Trace", section "JTAG interface"*.

2.4 Supported microcontrollers

The following table lists all the devices which are currently supported by Flasher RX.

Device	CPU core	Flash size [kByte]
R5F56104	RX610	768
R5F56106	RX610	1024
R5F56107	RX610	1536
R5F56108	RX610	2048
R5F56216	RX621	256
R5F56217	RX621	384
R5F56218	RX621	512
R5F562N7	RX62N	384
R5F562N8	RX62N	512
R5F562T6	RX62T	64
R5F562T7	RX62T	128
R5F562TA	RX62T	256

Table 2.2: Supported microcontrollers

2.5 Support of external flashes

Currently, programming of external NOR flash is not supported by Flasher RX. Thi limitation will be lifted in the near future.

2.6 Supported cores

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

- RX610
- RX621
- RX62N
- RX62T

Chapter 3

Remote control

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

3.1 Overview

There are 3 ways to control Flasher RX operation:

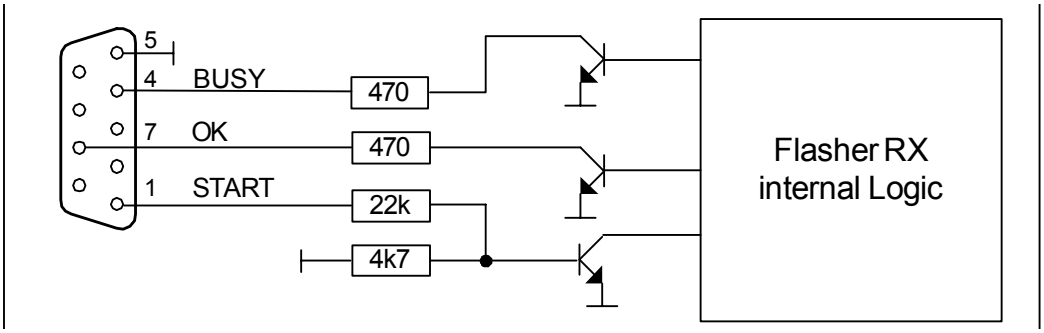
- Manual: Programming operation starts when pressing the button. The LEDs serve as visible indication.
- Via Handshake lines: 3 lines on the serial interface are used.
  1 line is an input and can be used to start operation,
  2 lines are outputs and serve as Busy and status output
- Terminal communication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

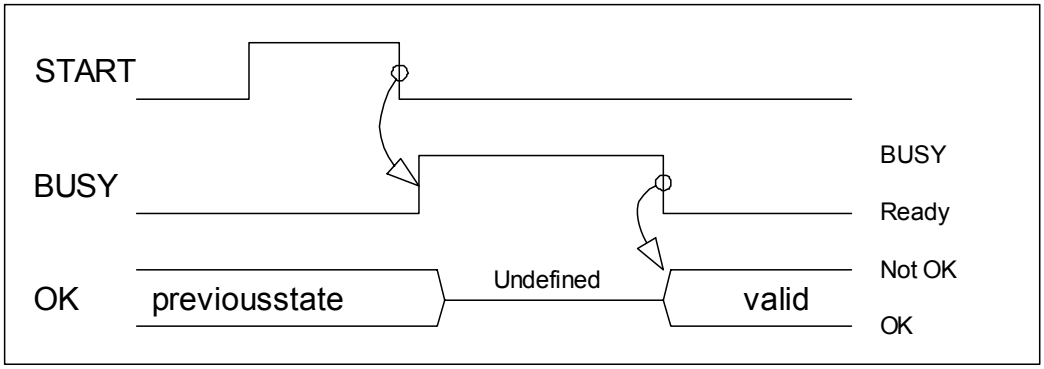
Note: All three ways to control Flasher RX operation are working only if Flasher RX is in standalone mode. In J-Link / MSD mode they have no effect.

3.2 Handshake control

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





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

Table 3.1: Flasher RX LED status

3.3 ASCII command interface

3.3.1 Introduction

Once set up using J-Flash, Flasher RX 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`.

3.3.2 General command and reply message format

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

3.3.3 Communication port settings

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

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

at 9600 baud.

3.3.4 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 J-Flash, this sequence can differ from the one shown above.

Finally, Flasher responds with

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

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

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


#AUTO 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 RX* on page 30.

#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.

3.3.4.1 File I/O commands

The ASCII interface of Flasher RX also supports file I/O operations via RS232. 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 send 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 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
```

#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
```

#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
```

#FREAD <Offset>,<NumBytes>

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

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

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

If the #FREAD command succeeds, Flasher will finally respond with a #OK:<NumBytes>:<Data> reply message. For more information about the Flasher reply messages, please refer to *Reply from Flasher RX* on page 30.

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

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

#FSIZE

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

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

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


If the #FSIZE command succeeds, Flasher will respond with a #OK:<Size> reply message. For more information about the Flasher reply messages, please refer to *Reply from Flasher RX* on page 30.

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
```

3.3.5 Reply from Flasher RX

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. <NumBytes> is the number of bytes which could be read. This value may differ from the number of requested bytes, for example if more bytes than available, were requested. <NumBytes> and <Data> are send in hexadecimal format (for <Data>: two hexadecimal characters per byte).

#OK:<Size>

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

#STATUS:

Flasher replies with its current state.

The following status messages are currently defined:

Message	Description
#STATUS:READY	Flasher is ready to receive a new command.
#STATUS:CONNECTING	Flasher initializes connection to target CPU.
#STATUS:INITIALIZING	Flasher performs self check and internal init.
#STATUS:UNLOCKING	Unlocking flash sectors.

Table 3.2: List of status messages that are currently defined

Message	Description
#STATUS:ERASING	Flasher is erasing the flash of the target device.
#STATUS:PROGRAMMING	Flasher is programming the flash of the target device.
#STATUS:VERIFYING	Flasher verifies the programmed flash contents.

Table 3.2: List of status messages that are currently defined

#ERRxxx

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

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

For example:

```
#ERR007:CANCELED.
```

The error code numbers are described in the following table:

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

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

Chapter 4

Performance

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

4.1 Performance of MCUs with internal flash memory

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

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

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

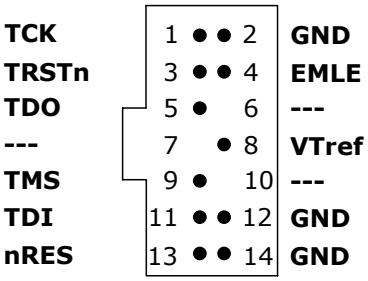
Chapter 5

Hardware

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

5.1 JTAG Connector

Flasher RX itself has a 20-pin JTAG connector mounted but comes with a 14-pin adapter for Renesas RX devices. This adapter also enables Flasher RX to optionally power the connected target hardware. On the adapter there is a jumper which allows selection between 3.3V and 5V supply target voltage supply. The target is supplied via the VTref connection when the supply option is jumpered.



5.1.1 Pinout

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

Table 5.1: Flasher RX pinout

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

5.1.2 Target board design for JTAG

We strongly advise following the recommendations given by the chip manufacturer. These recommendations are normally in line with the recommendations given in the table *Pinout* on page 36. In case of doubt you should follow the recommendations given by the semiconductor manufacturer.

5.1.2.1 Pull-up/pull-down resistors

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

5.1.3 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 300mA. 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"

Table 5.2: Command List

5.2 RESET, nTRST

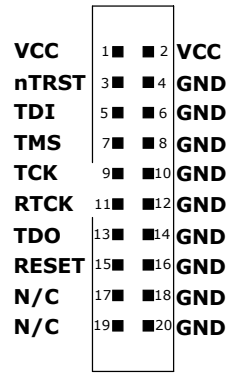
The TAP controller is reset independently from the RX core with nTRST. For the core to operate correctly it is essential that both signals are asserted after power-up.

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 address 0xFFFFFFFC, to allow the code to be single-stepped as soon as it comes out of reset). This can be particularly useful when first trying to bring up a board with a new ASIC.

5.3 Adapters

5.3.1 J-Link JTAG Isolator

The J-Link JTAG Isolator can be connected between Flasher RX and the 14-pin RX adapter which comes with the Flasher, 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 J-Link JTAG Isolator, please refer to *J-Link JTAG Isolator User Manual* (UM08010) which can be downloaded from our website.



5.3.2 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 RX 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. Not used on Renesas RX targets.
13	TDO	Input	JTAG data output from target CPU. Typically connected to TDO of target CPU.
15	RESET	I/O	Target CPU reset signal. Typically connected to the RESET pin of the target CPU, which is typically called "nRST", "nRESET" or "RESET".
17	N/C	N/C	This pin is not connected on the target side of the isolator.
19	N/C	N/C	This pin is not connected on the target side of the isolator.

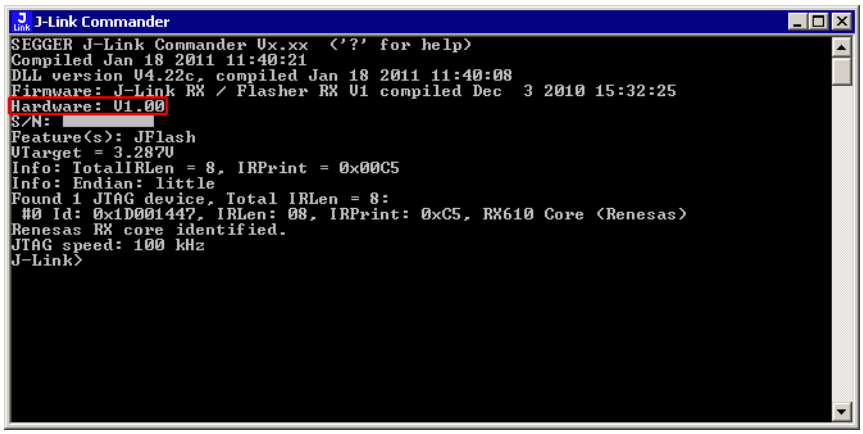
Table 5.3:

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

5.4 How to determine the hardware version

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

If this is not the case with your Flasher RX, you can use `JLink.exe` to determine your hardware version (if Flasher RX 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 RX is in J-Link mode, please refer to *J-Link mode* on page 12.



Chapter 6

Background information

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

6.1 Flash programming

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

6.1.1 How does flash programming via Flasher RX work ?

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

6.1.2 Data download to RAM

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

6.1.3 Available options for flash programming

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

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

6.1.3.1 Using Flasher RX in stand-alone mode

In order to use the Flasher RX in stand-alone mode, it has to be configured first. For more information about how to setup Flasher RX for using in "stand-alone mode", please refer to *Setting up Flasher RX for stand-alone mode* on page 13.

6.1.3.2 J-Flash - Complete flash programming solution

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

Chapter 7

Support and FAQs

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

7.1 Contacting support

Before contacting support, make sure you tried to solve your problem by trying your Flasher RX 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 RX.

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

- A detailed description of the problem
- Flasher RX serial number
- Information about your target hardware (processor, board, etc.).
- `FLASHER.CFG`, `FLASHER.DAT`, `FLASHER.LOG`, `SERIAL.TXT` file from Flasher RX. To get these files, Flasher RX has to be in MSD mode. For more information about how to boot Flasher RX in MSD mode, please refer to *MSD mode* on page 16.

Flasher RX is sold directly by SEGGER.

7.2 Frequently Asked Questions

Maximum JTAG speed

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

Maximum download speed

Q: What is the maximum download speed?
A: The maximum download speed is currently about 720 Kbytes/second when downloading into RAM. The actual speed depends on various factors, such as JTAG, clock speed, host CPU core etc.

Chapter 8

Glossary

This chapter describes important terms used throughout this manual.

Big-endian

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

Cache cleaning

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

Coprocessor

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

Dirty data

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

Halfword

A 16-bit unit of information.

Host

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

ICache

Instruction cache.

ID

Identifier.

IEEE 1149.1

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

Image

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

Instruction Register

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

IR

See Instruction Register.

Joint Test Action Group (JTAG)

The name of the standards group which created the IEEE 1149.1 specification.

Little-endian

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

Memory coherency

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

Memory management unit (MMU)

Hardware that controls caches and access permissions to blocks of memory, and translates virtual to physical addresses.

Memory Protection Unit (MPU)

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

RESET

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

nTRST

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

Open collector

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

Processor Core

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

Remapping

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

RTOS

Real Time Operating System.

TAP Controller

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

Target

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

TCK

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

TDI

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

TDO

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

Test Access Port (TAP)

The port used to access a device's TAP Controller. Comprises TCK, TMS, TDI, TDO, and nTRST (optional).

Transistor-transistor logic (TTL)

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

Word

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

Chapter 9

Literature and references

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

Reference	Title	Comments
[J-Link]	J-Link / J-Trace User Guide	This document describes J-Link and J-Trace. It is publicly available from SEGGER (*www.segger.com*).
[J-Flash]	J-Flash User Guide	This document describes J-Flash. It is publicly available from SEGGER (*www.segger.com*).

Table 9.1: Literature and References

Index

