

FF_RH850 and FF_RH850_UART Module Documentation

CSM GmbH, Filderstadt, Germany
www.csm.de/unicom/

July 2, 2024

Date	Version	Name	Changes
2014-07-02	1.0	CSM/RN	first release
2015-03-03	1.1	CSM/RN	FASTFLASH over CAN, DOWNLOAD, CONFIG_MODULE
2015-03-20	1.2	CSM/RN	MERGE and other new commands
2015-03-27	1.3	CSM/RN	SRECORD and INTELHEX file support, 16-byte-align for data flash programming
2015-04-14	1.4	CSM/RN	ICU S validation
2015-04-27	1.5	CSM/RS	adjust ICU S validation command, add ICU region erase and Read Data command
2015-05-19	1.6	CSM/RS	add new form in DOWNLOAD command
2015-07-15	1.7	CSM/RS	adjust response in ICU_S_VALIDATE, erase and blankcheck
2015-07-21	1.8	CSM/RS	add alternative FLMD0 and DSDI/TDO mapping for second controller programming
2017-05-18	1.9	CSM/RS	adjust READ_DATA and ERASE command
2017-10-25	2.0	CSM/RN	ext option byte, icu s option byte
2017-11-07	2.1	CSM/RN	switch verify command
2018-04-24	2.2	CSM/RN	RESET, SER_DISABLE, PROTECTION_BYTE added, READ_DATA, ERASE, BLANKCHECK, ID expanded
2018-05-25	2.3	CSM/RN	extended form of DEVICE_INFO command
2018-12-10	2.4	CSM/RN	Byte order option byte
2019-04-02	2.5	CSM/RN	ff_rh850_uart
2019-06-06	2.6	CSM/RN	GEN_IMAGE, SET_MERGE with multiple merge blocks
2019-09-30	2.7	CSM/RN	ICU_M_CHECK command
2020-03-20	2.8	CSM/BT	description of BLANK_CHECK updated
2020-09-10	2.9	CSM/BT	description of ICU_S_VALIDATE updated
2020-11-26	3.0	CSM/RN	removing of protection bits corrected
2021-04-14	3.1	CSM/RN	SCAN mode
2021-04-23	3.2	CSM/RN	AUTO mode
2022-09-20	3.3	CSM/RN	error codes added
2022-11-23	3.4	CSM/RN	Ethernet CRC32 with CHECKSUM_FILE
2024-07-02	3.5	CSM/RN	error codes corrected

All concepts and procedures introduced in this document are intellectual properties of CSM GmbH. Copying or usage by third parties without written permission of CSM GmbH is strictly prohibited. All trademarks mentioned in this document are properties of their respective owners. **This document is subject to changes without notice!**



CSM GmbH Computer-Systeme-Messtechnik
Raiffeisenstrasse 36 70794 Filderstadt-Bonlanden
Phone ++49 711 77964 0 Fax ++49 711 77964 40
mailto:unicom@csm.de <http://www.csm.de>

Copyright © 2021 by CSM GmbH

Contents

1	Introduction	5
2	Overview	6
3	Flash Mapping of RH850	7
4	SCAN and AUTO Mode	8
5	Loading and Configuration	9
5.1	MODULE Command	9
5.2	CONFIG_MODULE Command	12
6	FASTFLASH	14
7	FF_RH850 Module Commands	16
7.1	FF_RH850::ENTER_PROGRAMMING_MODE (1)	16
7.2	FF_RH850::READ_VERSION (2)	19
7.3	FF_RH850::DEVICE_INFO (3)	20
7.4	FF_RH850::GET_LAST_STATE (4)	22
7.5	FF_RH850::SET_MERGE (5)	23
7.6	FF_RH850::GET_SIGNATURE (6)	25
7.7	FF_RH850::GET_FLASH_AREA (7)	26
7.8	FF_RH850::PROTECTION_BYTE_COMMAND (8)	27
7.9	FF_RH850::DOWNLOAD (22)	29
7.10	FF_RH850::CHECKSUM (74)	31
7.11	FF_RH850::CHECKSUM_FILE (76)	32
7.12	FF_RH850::GEN_IMAGE (77)	35
7.13	FF_RH850::ICU_M_CHECK (105)	37
7.14	FF_RH850::ICU_S_VALIDATE (106)	38
7.15	FF_RH850::ICU_REGION_ERASE (107)	40
7.16	FF_RH850::ICU_S_OPTION_BYTE (108)	41
7.17	FF_RH850::OPTION_BYTE (109)	42
7.18	FF_RH850::EXT_OPTION_BYTE (110)	44

7.19	FF_RH850::CRC (111)	46
7.20	FF_RH850::READ_DATA (113)	47
7.21	FF_RH850::ERASE (115)	48
7.22	FF_RH850::PROGRAM (116)	50
7.23	FF_RH850::CLEAR_CONFIG (117)	51
7.24	FF_RH850::ID (118)	52
7.25	FF_RH850::SER_DISABLE (119)	54
7.26	FF_RH850::BLANKCHECK (124)	55
7.27	FF_RH850::SWITCH_VERIFY (126,127)	57
7.28	FF_RH850::RESET (255)	58
7.29	FF_RH850::ErrorCodes	59

Chapter 1

Introduction

FF_RH850 and FF_RH850_UART are modules for extension of *UCBASE* software running on UNICOM. The modules realize communication over the CSI respective UART interface of a *RENESAS RH850* micro controller for handling the internal flash memory of that device. Furthermore, they implement FASTFLASH over one or two CAN buses.

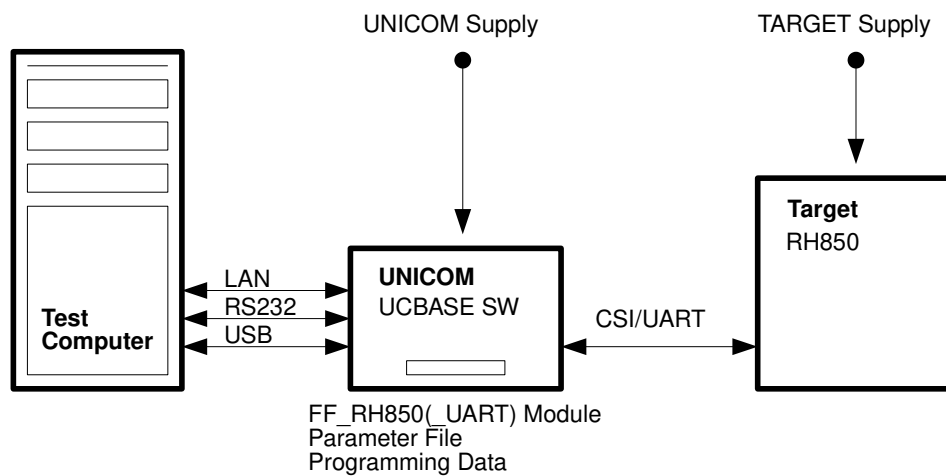
The FF_RH850 module uses the CSI interface of RH850, FF_RH850_UART respectively the UART interface. All commands of both modules are identical, so, in the following document, only FF_RH850 is mentioned.

Chapter 2

Overview

To use UNICOM device with FF_RH850 module, UNICOM must have at least hardware revision "C", and UCBASE software version 3.14 (!!) or newer must be installed on UNICOM.

The figure below shows the components that are basically needed, and there connections.



Chapter 3

Flash Mapping of RH850

The internal flash memory of RH850 is divided into

- User Flash
- User Boot Flash
- Data Flash

Each of the Flashes consist of *flash blocks* of different size which are separately erasable. User and User Boot Flash can be written page wise (each page is 256 bytes in size), Data Flash can be written in 16-byte-entities.

See table below for the exact mapping of internal flash memory.

Flash Type	Block	Start Address	Size
User	0	0x00000000	8kB
User	1	0x00002000	8kB
...			
User	7	0x0000E000	8kB
User	8	0x00010000	32kB
User	9	0x00018000	32kB
...			
User	21	0x00078000	32kB
User Boot	0	0x01000000	32kB
Data	0	0xFF200000	64 bytes
Data	1	0xFF200040	64 bytes
Data	2	0xFF200080	64 bytes
...			
Data	511	0xFF207FC0	64 bytes

Chapter 4

SCAN and AUTO Mode

The *SCAN mode* allows the FF_RH850 module to recognize the derivative of uC which is being connected to UNICOM, without specifying a parameter file. This can be used in the case that different uCs could be mounted on the current target device type. The result of SCAN mode can be used to build the name of parameter file for the uC type which has been found.

Starting the module in SCAN mode is simply done by skipping the parameter file name (an empty string consisting of eos(0) only must be specified with the MODULE command, ref chapter 5.1 on page 9).

After that, the ENTER_PROGRAMMING_MODE(1) command must be executed (ref. chapter 7.1 on page 16). If SCAN mode is active, this command reports the silicon signature with the name of the uC which has been recognized, with its response telegram.

As long as module is in SCAN mode, it can't execute commands that are dealing with the flash memory. Thus, after recognizing the uC type, the module must be re-started with its normal command form (with parameter file) in order to enable these commands again.

The *AUTO mode* is the consequent continuation of the SCAN mode approach. In this mode, the FF_RH850 module performs a scan and selects the matching parameter file by itself. No extra run of ENTER_PROGRAMMING_MODE must be executed.

To start the module in AUTO mode, a path name must be specified instead of a name of parameter file. This path must point to the pool of parameter files on UNICOM's storage medium. In order to distinguish the path name from a parameter file name, it must end with '`\`' (backslash, path delimiter).

Chapter 5

Loading and Configuration

5.1 MODULE Command

This command downloads and runs the FF_RH850 module.

Command, form 1 (unload module)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	0xC0	20	

Command, form 2 (load module, with CSI parameters)

byte 0	byte 1	byte 2	byte 3	...	byte xx	byte xx	
len	ecu	cmd	mod 1	...	mod n	eos	
N=n+m+9	0xC0	20		...		0	

byte xx	...	byte N-6	byte N-5	
par 1	...	par m	eos	
	...		0	

byte N-4	byte N-3	byte N-2	byte N-1	byte N
freq				cks
m1	m2	m3	e	

Command, form 3 (load module, no CSI parameters)

byte 0	byte 1	byte 2	byte 3	...	byte N-2	byte N-1	byte N
len	ecu	cmd	mod 1	...	mod n	eos	cks
N=4+n	0xC0	20		...		0	

len length of telegram

ecu	target address
cmd	command code
mod	filename of FF_RH850 module
eos	end-of-string (0)
par	name of parameter file
eos	end-of-string (0)
freq	oscillator frequency, see remarks
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	0xC0		

len	length of telegram
ecu	source address
status	result status
cks	checksum of telegram

Remarks

- After successful execution of the **MODULE** command, at least one interface slot must be configured for *MODULE* interface using the *UCBASE::-CONFIG_UNICOM(1)* command.
- The *parameter file* must match with the type of RH850 micro controller and the oscillator frequency.
- The *freq* parameter consists of a 4-byte-value that specifies the oscillator frequency in Hz. First three bytes build a three-digit-value that specifies the mantissa of the oscillator frequency, last byte specifies the number of zeros that have to be appended (exponent).
Example: 8MHz = 8000000Hz = 8 0 0 4
Example: 16 MHz = 16000000Hz = 1 6 0 5
- Form 3 of the command can be used if no CSI communication is necessary, for using FASTFLASH over CAN and the *load_sub* form of **DOWNLOAD** command (ref. chapter 7.9 on page 29).
- If *par* is an empty string (consisting only of eos), FF_RH850 module is being started in SCAN mode, refer to chapter 4 on page 8.
- If *par* is a path name (must end with '\'), FF_RH850 module is being started in AUTO mode, refer to chapter 4 on page 8. The path must point to the di-

rectory of UNICOM's storage medium, where parameter files can be found.
e.g. if the current working directory should be used as parameter file pool,
path must be ".\".

5.2 CONFIG_MODULE Command

This command controls whether FF_RH850 uses CAN or CSI interface for FASTFLASH end defines what type of file with programming data should be supported.

Command, form 1: FASTFLASH over CSI

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5
len	ecu	cmd	slot	config	cks
5	0xC0	4	0..3	0xX0	

Command, form 2: FASTFLASH over CAN

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6
len	ecu	cmd	slot	config	cans	cks
6	0xC0	4	0..3	0xX1	1,2	

len	length of telegram
ecu	target address
cmd	command code
slot	interface slot where MODULE interface is configured
config	<ul style="list-style-type: none"> • Low Nibble: 0: FASTFLASH over CSI (default), 1: FASTFLASH over CAN • High Nibble: 0: BINARY file (default), 4: SRECORD file, 8: INTELHEX file
cans	(only with config = 0xX1) number of used CANs: 1: CAN1, 2: CAN1+CAN2
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	0xC0		

len	length of telegram
ecu	source address
status	result status
cks	checksum of telegram

Remarks

- The default setting is FASTFLASH over CSI, BINARY file support.

- For FASTFLASH over CSI, the *ENTER_PROGRAMMING_MODE(1)* command (ref. chapter 7.1 on page 16) must have been successfully executed before.
- If SRECORD or INTELHEX is selected, the files must match with the flash granularity and alignment: code flash: 256-byte alignment, data flash: 16-byte-alignment. If the files contain blocks with not matching alignment, an ADDRESS_ERROR(0xB7) is reported while FASTFLASH is executed.

Chapter 6

FASTFLASH

FF_RH850 realizes *FASTFLASH* thru CSI interface, as well as FASTFLASH over CAN. Format of programming data must be *BINARY*, *SRECORD* or *INTELHEX*.

As well as the *X_FASTFLASH(14)* command and the *X_FASTFLASH_MULTI(15)* command is supported by the module.

Behaviour of FASTFLASH commands depend on the selected mode:

- *Program Mode*: The data which is stored in the file is being programmed into flash memory of RH850.
- *Verify Mode*: The data which is stored in the file is being compared against this one which are programmed into the flash memory of uC.
- The default mode is Program Mode. Mode selection can be done by using the SWITCH_VERIFY(126) command (ref. chapter 7.27 on page 57).

Generic FASTFLASH process looks like this:

- The module file, the parameter file and the flash programming data must reside on UNICOM's storage medium. If FASTFLASH over CAN should be used, a CAN operating system and a flash programming toolbox are necessary additionally. To copy these files to the storage medium, the file commands of the UCBASE software can be used or a tool that generates such commands automatically (e.g. *UniCMD*). This must be done only once until there are changes in flash programming data.
- The *CONFIG_MODULE(4)* command must be used to select between CSI interface and CAN, and to choose the type of file with the programming data. (ref. chapter 5.2 on page 12).
- Assuming that *FF_RH850* module is running on UNICOM and CSI communication between UNICOM and target is active if it should be used for

FASTFLASH. Further, the "MODULE interface" must be activated at least at one interface slot using the `CONFIG_UNICOM(1)` command of UCBASE software.

- Using the CSI interface, the programming mode of target must be activated by using the `ENTER_PROGMODE(1)` command.
- Flash memory should be checked whether it is empty (`BLANK_CHECK(124)`). If not blank, it must be erased using the `ERASE(115)` command before programming.
- Now, test computer sends an `X_FASTFLASH(14)` or `X_FASTFLASH_MULTI(15)` command to UNICOM. With these commands, it specifies the flash range(s) that is/are to be programmed, and the offset in the file that contains the flash programming data. Once received, UNICOM (FF_RH850 module) sends a sequence of flash programming commands to the target device using the flash programming data residing on the storage medium. No further actions of test computer are necessary.
- On UNICOM side, reading data from file and sending over CSI and on target's side, receiving over CSI and programming of the received data takes place in parallel. That ensures maximum transfer and programming rate.
- If all the flash programming data has been transferred, UNICOM sends a response telegram to the test computer.
- After programming, the flash data should be checked using either the `CHECKSUM(74)` or the `CRC(111)` command.

Please refer to `ucbase.pdf` for more information about FASTFLASH.

Please refer to `rh850_f11_os.pdf` for more information using the CAN buses for FASTFLASH.

Chapter 7

FF_RH850 Module Commands

7.1 FF_RH850::ENTER_PROGRAMMING_MODE (1)

This command let RH850 uC enter its programming mode. It must be executed prior any other target uC and CSI related commands.

Command (form 1, without ID)

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	
len	ecu	cmd	alt	dummy	br	
9	xx	1		0		

byte 6	byte 7	byte 8	byte 9
dummy	dummy	dummy	cks
0	0	0	

Command (form 2, with ID)

byte 0	byte 1	byte 2	
len	ecu	cmd	
25	xx	1	

byte 3	byte 4	byte 5	byte 6	byte 7	byte 8	
alt	dummy	br	dummy	dummy	dummy	
	0		0	0	0	

byte 9	...	byte 24	byte 25
id 0	...	id 15	cks
	...		

len length of telegram
ecu target address

cmd	command code
alt	alternative target interface mapping (not UART variant), see remarks
dummy	not used here, should be 0
br	CSI/UART bitrate in tenth of MBits/s (CSI: up to 5 MBits/s, UART: 500kb/s, 1 MBits/s, 1.5 MBits/s, 2 MBits/s). 0 selects default bitrate of 1 MBits/s
id	16 bytes of id code that protects serial programming if enabled.
cks	checksum of telegram

Response (success)

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

Response (wrong silicon signature, or scan/auto mode)

byte 0	byte 1	byte 2	byte 3	...	byte N-1	byte N
len	ecu	status	sig 1	...	sig n	cks
N=3+n	xx			...		

len	length of telegram
ecu	source address
status	result status
sig	in case of signature error, this field contains the real signature of the current RH850 uC. Most reason is an unmatching parameter file. Contact CSM for providing the right one.
cks	checksum of telegram

Remarks

- If FF_RH850 module has been started in SCAN or AUTO mode, the command always reports the silicon signature of uC which was recognized. Refer to chapter 4 on page 8.
- With the *alt* parameter, different GPIO mappings can be selected:

Alt Code	GPIO Mapping
0x00	GPIO1: RESET GPIO2: FLMD0 GPIO3: TCK GPIO4: TDI GPIO5: TDO
0x80	GPIO1: RESET GPIO2: FLMD0 GPIO3: TCK GPIO4: TDI GPIO5: TDO GPIO8: perm. high
0x81	GPIO1: RESET GPIO8: FLMD0 GPIO3: TCK GPIO4: TDI GPIO6: TDO GPIO2: perm. high
0x82	GPIO1: RESET GPIO2: FLMD0 GPIO3: TCK GPIO4: TDI GPIO7: TDO

7.2 FF_RH850::READ_VERSION (2)

This command reports about the version information of FF_RH850.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	2	

len length of telegram
ecu target address
cmd command code
cks checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	...	byte 18	byte 19
len	ecu	status	ver 1	...	ver 16	cks
19	xx					

len length of telegram
ecu source address
status result status
ver 1..16 version string
cks x checksum of telegram

Remarks

- As version string `ff_rh850_...Vx.yy` should be reported.

7.3 FF_RH850::DEVICE_INFO (3)

This command reports the device info string of the uC, or frequency values.

Command, standard form

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	3	

Command, extended form

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	sel	cks
4	xx	3	0,1	

len length of telegram
ecu source address
status result status
sel 0: info string is reported
 1: frequency values are reported
cks checksum of telegram

Response, standard or sel = 0

byte 0	byte 1	byte 2	byte 3	...	byte 26	byte 27
len	ecu	status	dev1	...	dev23	cks
27	xx			...		

Response, sel = 1

byte 0	byte 1	byte 2	byte 3	...	byte 6
len	ecu	cmd	CPU freq		
11	xx	3	MSB	...	LSB

byte 7	...	byte 10	byte 11
Per freq			cks
MSB	...	LSB	

len length of telegram
ecu source address
status result status
dev x device info
CPU freq CPU clock frequency in Hz

Per freq cks	Clock frequency for peripheral parts inside the uC in Hz checksum of telegram
-------------------------	--

7.4 FF_RH850::GET_LAST_STATE (4)

This command reports the last received status of the RH850 micor controller. It can be used in case of occurence of a *STATUS_ERROR* (0xEA) to figure out the reason.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	4	

len	length of telegram
ecu	target address
cmd	command code
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5
len	ecu	status	state	code	cks
5	xx				

len	length of telegram
ecu	source address
status	result status
state	reported state
code	reported error code
cks	checksum of telegram

Remarks

- Interpretation of reported status and error code must be done using the programming manual of the micro controller. It should only be used for debugging purpose.

7.5 FF_RH850::SET_MERGE (5)

That command enables the possibility to merge a small amount of data with the flash data to be programmed via FASTFLASH. The FASTFLASH process checks whether the address which is given with the SET_MERGE command is reached. If so, it substitutes the data of flash file against that one which are defined by SET_MERGE. Up to 8 amounts of merging data and a total data amount of 16384 bytes are supported.

Command, form 1, define merge data

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	opt	
N=8+n	xx	5	0	

byte 4	byte 5	byte 6	byte 7	
addr				
MSB			LSB	

byte 8	...	byte N-1	byte N
data 1	...	data n	cks
	...		

Command, form 2, enable/disable

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	enable	cks
4	xx	5	0/1/2	

len	length of telegram
ecu	target address
cmd	command code
opt	not used here, should be 0
addr	start address in flash where the merge data are to be programmed
data X	data bytes which have to be substituted against the data in flash file
enable	2: clear/reset merging data 1: merging while FASTFLASH enabled 0: merging while FASTFLASH disabled (default)
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len length of telegram
ecu source address
status result status
cks checksum of telegram

Remarks

- After startup of FF_RH850 module, the data merge is off. It should not be turned on by form 2 of command because no data and address have been given.
- Form 1 of command (defining merge data) enables merging automatically.
- The order of addresses of merge blocks never mind, however, ranges must not overlapp.

7.6 FF_RH850::GET_SIGNATURE (6)

This command reports the complete silicon signature of the RH850 micro controller.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	6	

len length of telegram
ecu target address
cmd command code
cks checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	...	byte 60	byte 61
len	ecu	status	sig 1	...	sig 58	cks
61	xx			...		

len length of telegram
ecu source address
status result status
sig silicon signature
cks checksum of telegram

7.7 FF_RH850::GET_FLASH_AREA (7)

This command reports information about the specified flash area.

Command

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	area	cks
4	xx	7		

len length of telegram
ecu target address
cmd command code
area Number of flash area that has to be recognized
cks checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	
len	ecu	status	type	start			
14	xx			MSB	...	LSB	

byte 8	...	byte 11	byte 12	byte 13	byte 14
blocksize			blocks		cks
MSB	...	LSB	MSB	LSB	

len length of telegram
ecu source address
status result status
type 0: Code Flash, 2: User Boot, 3: Data Flash
start start address of area
blocksize size of one flash block of area
blocks number of flash blocks of area
cks checksum of telegram

Remarks

- With this command can be figured out the flash size and the location.

7.8 FF_RH850::PROTECTION_BYTE_COMMAND (8)

This command reads or writes the protection byte of target.

Command (Read)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	8	

Command (Write)

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	prot_byte	cks
4	xx	8		

len length of telegram
ecu target address
cmd command code
prot_byte protection byte to write
cks checksum of telegram

Response (Read)

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	status	prot_byte	cks
4	xx			

len length of telegram
ecu source address
status result status
prot_byte protection byte read from target
cks checksum of telegram

Response (Write)

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

Remarks

- An ENTER_PROGRAMMING_MODE(1) command (ref. chapter 7.1 on page 16) must be executed before using this command.

- If bit 7 of *prot_byte* is programmed to 0, the flash read command is disabled (READ_DATA(113), ref. chapter 7.20 on page 47).
- If bit 6 of *prot_byte* is programmed to 0, the flash write command is disabled (PROGRAM(116), ref. chapter 7.22 on page 50, FASTFLASH).
- If bit 5 of *prot_byte* is programmed to 0, the flash erase command is disabled (ERASE(115), ref. chapter 7.21 on page 48).
- The protection bits can be removed by erasing the entire flash memory (ERASE(115), refer chapter 7.21 on page 48), followed by a CLEAR_CONFIG(117) command, (refer chapter 7.23 on page 51). It is not possible if the erase protection is enabled.

7.9 FF_RH850::DOWNLOAD (22)

With this command, a toolbox can be downloaded either using the CSI interface or STP.

Command (form 1, download over CSI, called *bootstrap*)

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7	
len	ecu	cmd	opt	addr				
N=9+n+m	xx	22	0	MSB			LSB	

byte 8	...	byte x	byte x	byte x	...	byte N-1	byte N
file 1	...	file n	EOS	param 1	...	param m	cks
	...		0		...		

Command (form 2, download over STP, called *load_sub*)

byte 0	byte 1	byte 2	byte 3	byte 4	
len	ecu	cmd	opt	slot	
N=6+n	xx	22	1	0..3	

byte 5	...	byte N-2	byte N-1	byte N
file 1	...	file n	EOS	cks
	...		0	

Command (form 3, download over STP with additional parameters)

byte 0	byte 1	byte 2	byte 3	byte 4	
len	ecu	cmd	opt	slot	
N=6+n+m	xx	22	1	0..3	

byte 5	...	byte x	byte x	
file 1	...	file n	EOS	
	...		0	

byte x	...	byte N-1	byte N
param 1	...	param m	cks
	...		

len	length of telegram
ecu	target address
cmd	command code
opt	0: bootstrap, 1: load_sub
addr	destination address in RAM (bootstrap only)
file x	name of toolbox file, must reside on UNICOM's storage medium
EOS	End-Of-String (0)
param x	optional additional params, depending on toolbox (bootstrap only)
slot	interface slot which is used for download. An STP interface (STPonCAN, STPonKLine...) must be configured on this slot (load_sub only)
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len	length of telegram
ecu	source address
status	result status
cks	checksum of telegram

Remarks

- If form 1 is used, the *ENTER_PROGRAMMING_MODE(1)* (ref. chapter 7.1 on page 16) command must have been executed successfully before.
- For using form 2 and 3, an STP able software must be running on target side.

7.10 FF_RH850::CHECKSUM (74)

This command computes a checksum over an entire flash area

Command

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	area	cks
5	xx	74		

len	length of telegram
ecu	target address
cmd	command code
area	flash area the checksum has to be computed: 0x00: code flash 1, 0x01: code flash 2, 0x10: user boot, 0x20: data flash
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	...	byte 6	byte 7
len	ecu	status	Checksum			cks
7	xx		MSB	...	LSB	

len	length of telegram
ecu	source address
status	result status
checksum	32bit sum of all the bytes contained by the selected area
cks	checksum of telegram

7.11 FF_RH850::CHECKSUM_FILE (76)

That command computes a checksum over a range of a given file with the same algorithm as RH850 controller it does (either NEC checksum or RH850 CRC). If merging data are defined and enabled (s. SET_MERGE(5), ref. chapter 7.5 on page 23), that data is taken into account for the checksum computation.

Command

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	opt	
N=16+n	xx	76	0,1,2	

byte 4	byte 5	byte 6	byte 7	
start				
MSB			LSB	

byte 8	byte 9	byte 10	byte 11	
end				
MSB			LSB	

byte 12	byte 13	byte 14	byte 15	
offset				
MSB			LSB	

byte 16	...	byte N-2	byte N-1	byte N
file 1	...	file n	EOS	cks
	...		0	

len	length of telegram
ecu	target address
cmd	command code
opt	0: NEC checksum, 1: RH850 CRC32, 2: CCITT CRC32
start/end	start and end address as in CRC(111) command
offset	offset in file
file	(optional) name of file that contains programming data, same as used for FASTFLASH
EOS	(optional) end-of-string (0), must be inserted if <i>file</i> is specified
cks	checksum of telegram

Response, NEC checksum

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5
len	ecu	status	checksum		cks
5	xx		MSB	LSB	

Response, CRC32

byte 0	byte 1	byte 2	byte 3	...	byte 6	byte 7
len	ecu	status	crc32			check
7	xx		MSB	...	LSB	

len	length of telegram
ecu	source address
status	result status
checksum	checksum computation result
cks	checksum of telegram

Remarks

- That command should be used in order to compute the expected checksum over the file with applied merge data. After programming via FASTFLASH, the CRC (111) command should report the same checksum as calculated with this command. *opt* must be specified as 1 (RH850 CRC) in this case.
- If no *file* is specified, DEFAULT.DAT is assumed.
- The *NEC Checksum* is the 2's complement of the 16 bit sum over all the bytes of file.
- Table on next page shows CRC parameters.

Name	opt	Width	Poly	Init	RefIn	RefOt	XorOt
RH850_CRC32	1	32	0x04C11DB7	0xFFFFFFFF	FALSE	FALSE	0x00000000
CCITT_CRC32	2	32	0x04C11DB7	0xFFFFFFFF	TRUE	TRUE	0xFFFFFFFF

7.12 FF_RH850::GEN_IMAGE (77)

This command generates a flash image file from specified programming data file with respect to the data blocks specified with the SET_MERGE(5) command (ref. chapter 7.5 on page 23).

Command

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	opt	
N=16+n+m	xx	77	0	

byte 4	byte 5	byte 6	byte 7	
start				
MSB			LSB	

byte 8	byte 9	byte 10	byte 11	
end				
MSB			LSB	

byte 12	byte 13	byte 14	byte 15	
offset				
MSB			LSB	

byte 16	...	byte x	byte x	
pfile 1	...	pfile n	EOS	
	...		0	

byte x	...	byte N-2	byte N-1	byte N	
ifile 1	...	ifile m	EOS	cks	
	...		0		

len	length of telegram
ecu	target address
cmd	command code
opt	not used here, should be 0
start/end	start and end address in flash memory
offset	offset in file
pfile	name of file that contains programming data, same as used for FASTFLASH
EOS	end-of-string (0)
ifile	name of resulting image file
EOS	end-of-string (0)
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len length of telegram
ecu source address
status result status
cks checksum of telegram

Remarks

- The main purpose of the command is checking whether the merge areas will be properly placed inside the flash memory by copying the resulting image file to PC for analyze.
- The command is not intended for creating flash image files on the fly with the normal production process.

7.13 FF_RH850::ICU_M_CHECK (105)

This command checks the state of ICU-M

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	105	

len length of telegram
ecu target address
cmd command code
cks checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	status	icu m	cks
N=4	xx			

len length of telegram
ecu source address
status result status
icu m 0x00: on, 0xFF: off, 0xAA: secured
cks checksum of telegram

Remarks

- ICU-M can be enabled by writing proper extended option bytes, refer EXT_OPTION_BYTE, chapter 7.18 on page 44

7.14 FF_RH850::ICU_S_VALIDATE (106)

This command whether validates the ICU S unit or checks if the ICU unit is valid/invalid.

Command, form 1 (validate)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	106	

Command, form 2 (check if valid or invalid)

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	exp	cks
4	xx	106	0x00/0xFF	

len length of telegram
ecu target address
cmd command code
exp expected valid byte
 0x00: valid, 0xFF: invalid
cks checksum of telegram

Response, form 1 (validate)

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len length of telegram
ecu source address
status result status
cks checksum of telegram

Response, form 2 (check)

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	status	result	cks
4	xx		0xA0/0xF7	

len length of telegram
ecu source address

status	result status
result	0xA0: ICU S in expected state 0xF7: RH850_VERIFY_ERROR (not in expected state)
cks	checksum of telegram

Remarks

- Not all RH850 devices have ICU S unit, thus, command could fail with STATUS_ERROR.
- When checking if ICU is valid (*expected valid byte* set to 0x00) and a RH850_VERIFY_ERROR is reported in the response telegram, the ICU S unit has not been validated yet.
- When checking for a non-validated ICU (*expected valid byte* set to 0xFF) and a RH850_VERIFY_ERROR is reported, the ICU S unit has already been validated.

7.15 FF_RH850::ICU_REGION_ERASE (107)

This command erases the ICU S unit region in the last 1kByte of targets data flash and invalidates the ICU-S valid flag.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	107	

len length of telegram
ecu target address
cmd command code
cks checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len length of telegram
ecu source address
status result status
cks checksum of telegram

Remarks

- Not all RH850 devices have ICU S unit, thus, command could fail with STATUS_ERROR.

7.16 FF_RH850::ICU_S_OPTION_BYTE (108)

With this command, a special option byte for the ICU S unit (OPT9) can be programmed. However, there is another command that can do this: EXT_OPTION_BYTE(110) (ref. chapter 7.18 on page 44). See remarks what command must be used with the current RH850 type.

Command

byte 0	byte 1	byte 2	byte 3	...	byte 6	byte 7
len	ecu	cmd	OPT9			cks
7	xx	108	LSB(!)	...	MSB(!)	

len	length of telegram
ecu	target address
cmd	command code
OPT9	option byte for ICU_S, LSB first!
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len	length of telegram
ecu	source address
status	result status
cks	checksum of telegram

Remarks

- Whether and which command can be used to programming of the OPT9 option byte depends on type of RH850. It can be figured out with the DEVICE_INFO(3) command (ref. chapter 7.3 on page 20). If bit 1 in the dev7 value is set to 1, the EXT_OPTION_BYTE(110) command (ref. chapter 7.18 on page 44) works, and if bit 3 is set to 1, the ICU_S_OPTION_BYTE(108) command (ref. chapter 7.16 on page 41) can be used. In all other cases, OPT9 can't be programmed.

7.17 FF_RH850::OPTION_BYTE (109)

With this command either the current option byte values can be recognised or new option byte values can be programmed.

Command (Read)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	109	

Command (Write)

byte 0	byte 1	byte 2	byte 3	...	byte 34	byte 35
len	ecu	cmd	option bytes			cks
35	xx	109		...		

len length of telegram
ecu target address
cmd command code
option bytes option bytes
cks checksum of telegram

Response (Read)

byte 0	byte 1	byte 2	byte 3	...	byte 34	byte 35
len	ecu	status	option byte			cks
35	xx			...		

Response (Write)

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len length of telegram
ecu source address
status result status
option bytes option bytes
cks checksum of telegram

Remarks

- In most cases the *option bytes* are specified as 32-bit words. If so, the least significant byte of these words must inserted first into the command tele-

gram. EXAMPLE: if the OPB0 is specified as *0xBFA4FFDF*, the command telegram for writing must look as follows:

```
35 <ecu> 109 0xDF 0xFF 0xA4 0xBF 0xFF ... 0xFF cks
```

7.18 FF_RH850::EXT_OPTION_BYTE (110)

With this command, some extra option bytes can be programmed and verified. Note that not all RH850 types support this command.

Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	...	byte 8	
len	ecu	cmd	opt	OPT8H	OPT10			
23	xx	110	0,1		LSB(!)	...	MSB(!)	

byte 9	...	byte 12	byte 13	...	byte 16	byte 17	byte 18	
OPT11			OPT9			OPT8M	OPT8L	
LSB(!)	...	MSB(!)	LSB(!)	...	MSB(!)			

byte 19	...	byte 22	byte 23
OPT12			cks
LSB(!)	...	MSB(!)	

len	length of telegram
ecu	target address
cmd	command code
opt	0: write, 1: verify
OPT8H	If OPT8 bits 28..31 = 1111b, OPT8H = 0xFF, else 0x00
OPT10	value for OPT10, LSB first!
OPT11	value for OPT11, LSB first!
OPT9	value for OPT9, LSB first!
OPT8M	If OPT8 bits 20..23 = 1111b, OPT8M = 0xFF, else 0x00
OPT8L	If OPT8 bits 12..15 = 1111b, OPT8L = 0xFF, else 0x00
OPT12	value for OPT12, LSB first!
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len	length of telegram
ecu	source address
status	result status
cks	checksum of telegram

Remarks

- Bit 1 of *dev1* value must be 1 if this command should work (ref. DEVICE_INFO(3), chapter 7.3 on page 20).
- Whether and which command can be used to programming of the OPT9 option byte depends on type of RH850. It can be figured out with the DEVICE_INFO(3) command (ref. chapter 7.3 on page 20). If bit 1 in the *dev7* value is set to 1, the EXT_OPTION_BYTE(110) command (ref. chapter 7.18 on page 44) works, and if bit 3 is set to 1, the ICU_S_OPTION_BYTE(108) command (ref. chapter 7.16 on page 41) can be used. In all other cases, OPT9 can't be programmed.

7.19 FF_RH850::CRC (111)

With this command, a 32bit CRC can be computed over flash ranges.

Command

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	opt	
12	xx	111	0	

byte 4	...	byte 7	byte 8	...	byte 11	byte 12
start_addr			end_addr			check
MSB	...	LSB	MSB	...	LSB	

len	length of telegram
ecu	target address
cmd	command code
opt	not used, should be 0
start_addr	start address of range, must be 1kByte aligned
end_addr	end address of range, end_addr+1 must be 1kByte aligned
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	...	byte 6	byte 7
len	ecu	status	crc32		check	
7	xx		MSB	...	LSB	

len	length of telegram
ecu	source address
status	result status
crc32	computed checksum
cks	checksum of telegram

Remarks

- Polynomial is 0x04C11DB7, start value is 0xFFFFFFFF

7.20 FF_RH850::READ_DATA (113)

With help of this command, data from code and data flash can be read.

Command

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	opt	
10	xx	113	0	

byte 4	...	byte 7	byte 8	byte 9	byte 10
start_addr			size		cks
MSB	...	LSB	MSB	LSB	

len	length of telegram
ecu	target address
cmd	command code
opt	dummy byte, has to be 0
start_addr	start address in data flash of target
size	size of read data
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	...	byte N-1	byte N
len	ecu	status	data_1	...	data_n	cks
N=3+n	xx					

len	length of telegram
ecu	source address
status	result status
data_x	read data from data flash of target
cks	checksum of telegram

Remarks

- An ENTER_PROGRAMMING_MODE(1) command (ref. chapter 7.1 on page 16) must be executed before using this command.
- Data can be read byte wise without any alignment rule.

7.21 FF_RH850::ERASE (115)

This command erases the flash memory of entire chip or e specified area. This includes the code, user boot and data flash of RH850. It also can skip a number of 64 byte blocks at the end of the data flash for handling the ICU of target (e.g. for disabling ICU Unit).

Command (form 1: bulk erase)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	115	

Command (form 2: erase and skipping blocks of data flash)

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	blocks	cks
4	xx	115		

Command (form 3: erase a flash area)

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	opt	
12	xx	115	0	

byte 4	...	byte 7	byte 8	...	byte 11	byte 12
start_addr			end_addr			check
MSB	...	LSB	MSB	...	LSB	

len	length of telegram
ecu	target address
cmd	command code
blocks	number of 64 byte blocks to be skipped while erasing data flash (only used for ICU handling)
start_addr	address of first flash block to erase
end_addr	address of last flash block to erase
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len	length of telegram
------------	--------------------

ecu	source address
status	result status
cks	checksum of telegram

Remarks

- An ENTER_PROGRAMMING_MODE(1) command (ref. chapter 7.1 on page 16) must be executed before using this command.
- All the recognised flash blocks are being erased after another.
- if erasing the whole data flash is blocked by ICU of target, the data flash has to be erased, except the last 1024 bytes. Using the *blocks* parameter (64byte blocks), memory at the end of data flash can be skipped (e.g. $0x10 = 16 * 64 \text{ byte} = 1024 \text{ bytes}$ will be skipped). Refer ICU_REGION_ERASE command (chapter chapter 7.15 on page 40) for more information.
- This command must be executed before programming of the flash memory if it is not blank. That can be checked by executing the BLANKCHECK(124) command (refer chapter 7.26 on page 55)
- If the area form is used, a contiguous flash range must be specified.

7.22 FF_RH850::PROGRAM (116)

This command programs multiple of 256 data bytes directly into the code flash memory (user and user boot), respectively multiple of 16 data bytes into the data flash memory of RH850 micro controller.

Command

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	
len	ecu	cmd	opt	addr			
N=8+n	xx	116	0	MSB	...	LSB	

	byte 8	...	byte N-1	byte N
	data 1	...	data n	cks
		...		

len	length of telegram
ecu	target address
cmd	command code
opt	not used, should be 0
addr	destination address in flash, must be 256-byte-aligned for code flash and 16-byte-aligned for data flash
data	data bytes to be programmed, must be multiple of 256 for code flash and multiple of 16 for data flash.
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len	length of telegram
ecu	source address
status	result status
cks	checksum of telegram

Remarks

- Since multiple of 256 data bytes must be specified with one command telegram, this command only works with *XSTP* mode activated.
- Flash memory must be in erased state before this command can be used.

7.23 FF_RH850::CLEAR_CONFIG (117)

The command clears configuration bits of RH850, e.g. the ID value.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	117	

len length of telegram
ecu target address
cmd command code
cks checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len length of telegram
ecu source address
status result status
cks checksum of telegram

7.24 FF_RH850::ID (118)

With this command, the actual programmed ID can be reported, or a new one can be programmed.

Command form 1 (read)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	118	

Command form 2 (program, without spie)

byte 0	byte 1	byte 2	byte 3	...	byte 18	byte 19
len	ecu	cmd	id 0	...	id 15	cks
19	xx	118	LSB	...	MSB	

Command form 3 (program, with spie)

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 18	byte 20
len	ecu	cmd	spie	id 0	...	id 15	cks
20	xx	118		LSB	...	MSB	

len	length of telegram
ecu	target address
cmd	command code
spie	<i>Serial Programming ID Enable</i> , 0: ID is only valid for flash else: serial programming is protected by ID
id	ID code to program (16 bytes in memory order)
cks	checksum of telegram

Response form 1 (read)

byte 0	byte 1	byte 2	byte 3	...	byte 18	byte 19
len	ecu	status	id 0	...	id 15	cks
19	xx		LSB	...	MSB	

Response form 2 (program)

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len	length of telegram
------------	--------------------

ecu	source address
status	result status
id	current ID code (16 bytes in memory order)
cks	checksum of telegram

Remarks

- ID code can be used to protect self programming via application software and protect serial programming mode.
- If spie is active, the programmed ID must be specified with the ENTER_PROGRAMMING_MODE(1) (ref. chapter 7.1 on page 16)
- It can be removed by erasing the entire flash memory (ERASE(115), ref. chapter 7.21 on page 48), followed by a CLEAR_CONFIG(117) command (ref. chapter 7.23 on page 51).

7.25 FF_RH850::SER_DISABLE (119)

This command disables the serial programming interface. !!CAUTION!! This can not be revoked !!

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	119	

len length of telegram
ecu target address
cmd command code
cks checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len length of telegram
ecu source address
status result status
cks checksum of telegram

Remarks

- The serial programming interface is disabled after the next reset.

7.26 FF_RH850::BLANKCHECK (124)

This command checks the entire flash memory of RH850 micro controller whether it is in erased state.

Command (form 1: entire chip)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	124	

Command (form 2: skipping blocks of data flash)

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	blocks	cks
4	xx	124		

Command (form 3: check a flash area)

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	opt	
12	xx	124	0	

byte 4	...	byte 7	byte 8	...	byte 11	byte 12
start_addr			end_addr			check
MSB	...	LSB	MSB	...	LSB	

len	length of telegram
ecu	target address
cmd	command code
blocks	number of 64 byte blocks to be skipped while checking data flash (only used for ICU handling)
start_addr	start of check
end_addr	end of check
cks	checksum of telegram

Response form 1: status only

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

Response form 2: status and address information

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7
len	ecu	status	addr				cks
7	xx		MSB			LSB	

len	length of telegram
ecu	source address
status	result status
addr	word (4-Byte) address of block where blank check failed
cks	checksum of telegram

Remarks

- An ENTER_PROGRAMMING_MODE(1) command (ref. chapter 7.1 on page 16) must be executed before using this command.
- For command form 1 and form 2, all flash blocks are checked one after another. If all blocks are blank, NO_ERROR (0xA0) is reported (response form 1). If a block is non-empty, status BLANKCHECK_ERROR (0xE5) is reported and the address of block start is returned (response form 2).
- For command form 3 (check a flash area), just the status is reported (response form 1).
- This command should be executed before programming of the flash memory. If it is not blank, it must be erased using the ERASE(115) command (refer chapter 7.21 on page 48)

7.27 FF_RH850::SWITCH_VERIFY (126,127)

With this command, FASTFLASH can be configured either to program data (*Program Mode*) or to verify them (*Verify Mode*). Verify mode should be used when programming data exists as SRECORD or INTEL file, and data is not contiguous so that verifying the flash data by checksum is not possible (e.g. with data flash).

Command

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	mode	cks
4	xx	126,127	0, 1	

len	length of telegram
ecu	target address
cmd	command code
mode	selected mode: 0: program mode (default), 1: verify mode
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len	length of telegram
ecu	source address
status	result status
cks	checksum of telegram

Remarks

- Verifying of flash data is done by selecting verify mode and repeating execution of the X_FASTFLASH(14) or X_FASTFLASH_TAB(15) command. Instead of programming the data, flash memory is being read and compared with data in file.
- This type of verify is slower then using the checksum method. It should only be used with small amount of data, if data exists as SRECORD or INTEL file and there are gaps in the address layout.

7.28 FF_RH850::RESET (255)

This command resets the connected target device and runs it in application mode.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	255	

len length of telegram
ecu target address
cmd command code
cks checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

len length of telegram
ecu source address
status result status
cks checksum of telegram

Remarks

- The command can be used to startup a just programmed application software.
- After executing the command, target uC is not longer in programming mode.

7.29 FF_RH850::ErrorCodes

The following table describes possible error codes reported by the *status* parameter of the response telegrams, and their meanings.

Error	Code	Description
NO_ERROR	0xA0	no error occurred
PARAMETER_ERROR	0xB0	wrong parameter in command telegram
LENGTH_ERROR	0xB3	wrong command telegram length
FILE_ERROR	0xB9	error opening or reading file
MERGE_ERROR	0xDF	error while defining merge data blocks
NO_VGPIO_ERROR	0xE0	no V_GPIO detected
NOT_IN_PROGMODE_ERROR	0xE1	uC is not in programming mode
M_PARAMFILE_ERROR	0xE2	wrong or corrupted parameter file
M_CLOCK_FREQ_MISMATCH	0xE3	clock frequency doesn't match with parameter file
M_WRONG_SILICON_SIGNATURE	0xE4	signature doesn't match with par-file
M_BLANKCHECK_ERROR	0xE5	flash not blank
M_TIMEOUT_ERROR	0xE6	no response from target within timeout time
M_FORMAT_ERROR	0xE7	wrong response from target
M_BUFFER_OVERFLOW	0xE8	reponse from target too large
M_CHECKSUM_ERROR	0xE9	wrong frame checksum from target
M_STATUS_ERROR	0xEA	wrong status from target
IN_SCAN_MODE_ERROR	0xEB	command failed since module is in SCAN mode
M_HS_ERROR	0xED	handshake error (target)
M_COM_ERROR	0xEE	communication error (target)
M_NOT_SUPPORTED_ERROR	0xEF	not supported feature used
FLASH_AREA_OVERFLOW_ERROR	0xF1	too much flash areas recognised
WRONG_FLASH_TYPE_ERROR	0xF2	unsupported flash area type recognised
ICU_VALDATE_ERROR	0xF3	ICU unit could not validated
VERIFY_ERROR	0xF4	Verifying of flash data in Verify Mode failed
UNSUPPORTED_FREQUENCY_ERROR	0xF5	Unsupported clock frequency specified (AUTO mode)
RH850_BLANK_ERROR	0xF6	Blank Check of a flash area failed
RH850_VERIFY_ERROR	0xF7	Valdiation of ICU S failed
UNKNOWN_COMMAND_ERROR	0xFF	command not supported

More error codes are described in `ucbase.pdf`