

FF_RL78F Module Documentation

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

May 25, 2021

Date	Version	Name	Changes
2014-03-20	1.0	CSM/RN	first release
2016-03-16	1.1	CSM/RS	EnterProgMode-add VOUTx switches SetMerge-add multiple Merge descr.
2016-09-22	1.2	CSM/RS	add CONFIG_INTERFACE command
2017-03-24	1.21	CSM/RN	CONFIG_INTERFACE command corrected
2017-05-30	1.3	CSM/RN	new MERGE, parametrizable CRC computation
2018-03-26	1.4	CSM/RN	GET_INFO command
2020-02-21	1.5	CSM/TO	configure reset pin
2020-03-17	1.6	CSM/RN	HEX files, verify mode
2021-04-20	1.61	CSM/TO	clarification SECURITY command
2021-05-05	1.7	CSM/RN	scan and auto mode
2021-05-25	1.71	CSM/RN	CHECK_FLASH command 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	3
2	Overview	4
2.1	Requirements	4
2.2	Connecting Target	5
2.3	Power Supply of Target	5
3	SCAN and AUTO Mode	7
4	Loading and Configuration	8
4.1	MODULE Command	8
4.2	CONFIG_INTERFACE Command	11
5	FASTFLASH	13
6	FF_RL78F Commands	14
6.1	FF_RL78F::ENTER_PROGMODE (1)	14
6.2	FF_RL78F::READ_VERSION (2)	16
6.3	FF_RL78F::GENERIC_COMMAND (3)	17
6.4	FF_RL78F::GET_LAST_STATE (4)	18
6.5	FF_RL78F::SET_MERGE (5)	19
6.6	FF_RL78F::CONFIG_CRC (6)	21
6.7	FF_RL78F::GET_INFO (7)	23
6.8	FF_RL78F::GET_SIGNATURE (8)	24
6.9	FF_RL78F::CHECKSUM (74)	25
6.10	FF_RL78F::CHECKSUM_FILE (76)	27
6.11	FF_RL78F::GEN_IMAGE (77)	29
6.12	FF_RL78F::SECURITY (110)	31
6.13	FF_RL78F::ERASE_FLASH (115)	33
6.14	FF_RL78F::SWITCH_VERIFY (117)	35
6.15	FF_RL78F::CHECK_FLASH (124)	36
6.16	FF_RL78F::ErrorCodes	38

Chapter 1

Introduction

FF_RL78F is a module for extending the *UCBASE* software running on *UNI-COM3*. It implements programming of flash memory of RENESAS RL78F (R5Fxxxx) based devices over its "TOOL0" interface.

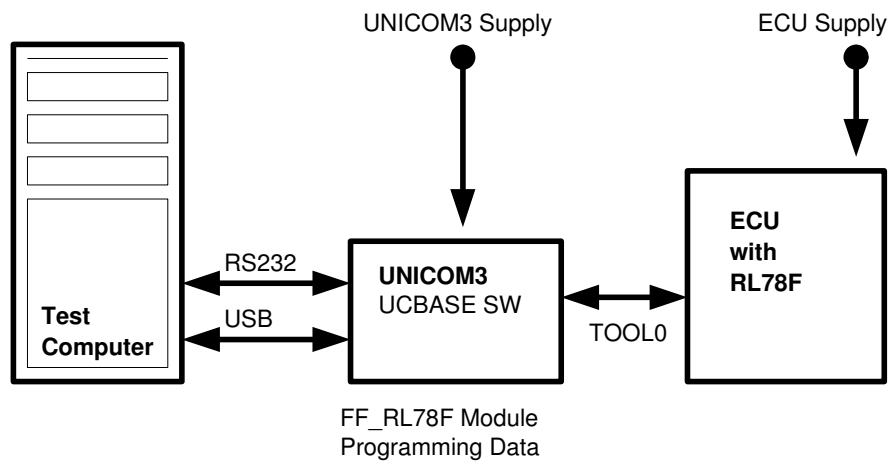
Chapter 2

Overview

2.1 Requirements

To use UNICOM3 device with FF_RL78F module, UCBASE software version 3.06 or newer must be installed on UNICOM3.

The figure below shows the components of the system.



2.2 Connecting Target

The figure below shows connecting the target device to UNICOM3 as a matter of principle. Please refer additional information about connection that comes with the project specific software delivery.

2.3 Power Supply of Target

There are 2 different possibilities to power the target device:

- Using the power supply of target itself
- Using UNICOM3 for power supply

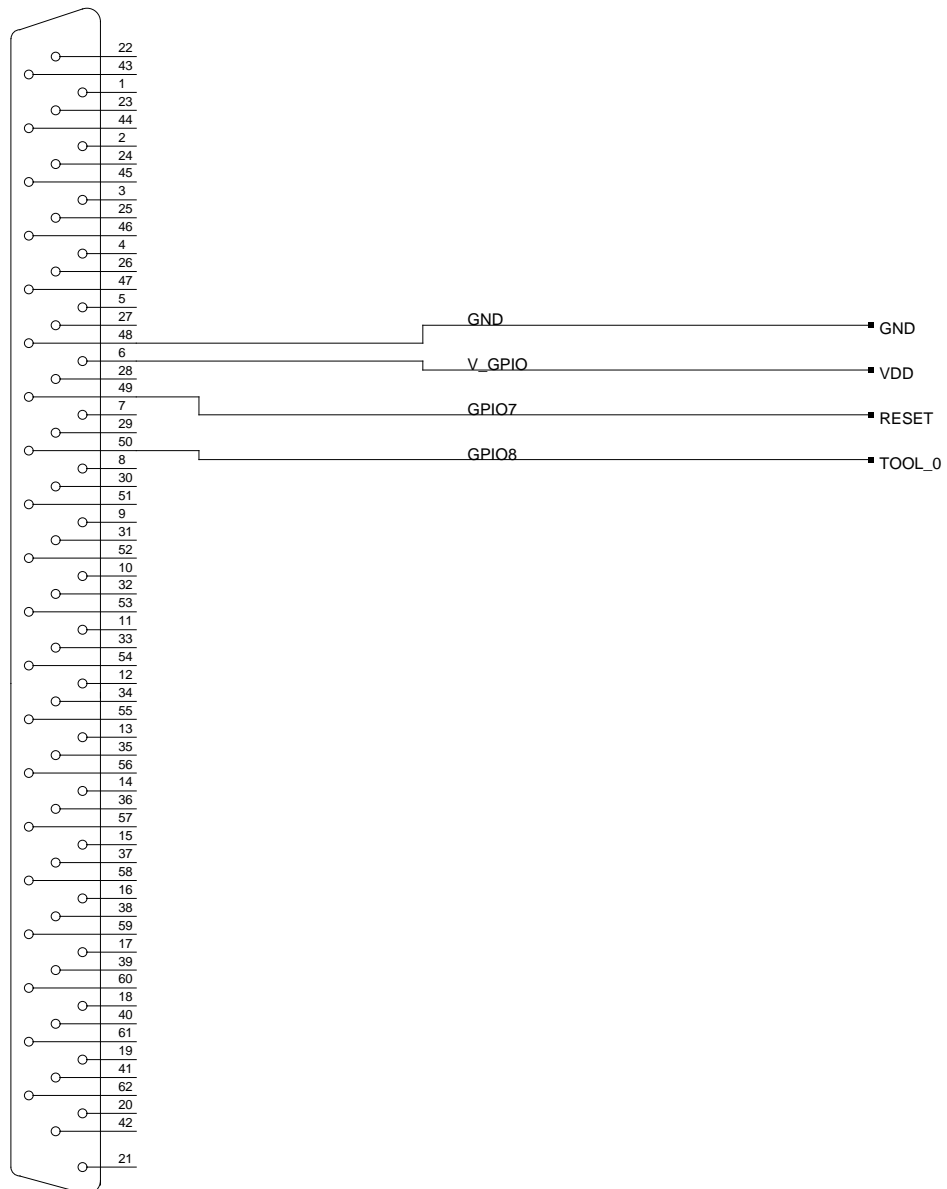
Using the first way, UNICOM3 must not provide any power supply at V_GPIO pin (DSUB62/6). This is the default from startup of UNICOM3. To ensure this, a `CONTROL_VGPIO(70)` command with `VGPI0` parameter set to 0 must be executed before target can be powered up. However, the pin must be connected to the VCC voltage of target device in order to power the GPIO driver stages of UNICOM3. This should be the normal way to connect the target.

If no target power supply is available, the second way can be used. In this case, a `CONTROL_VGPIO(70)` command with `VGPI0` parameter set to 33 or 50 according to the desired voltage value before loading of `FF_RL78F` module.

Refer to `ucbase.pdf` for more information about the `CONTROL_VGPIO` command.

UNICOM3
DSUB62

RL78F



Chapter 3

SCAN and AUTO Mode

The *SCAN mode* allows the FF_RL78F 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 4.1 on page 8).

After that, the ENTER_PROGRAMMING_MODE(1) command must be executed (ref. chapter 6.1 on page 14). 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_RL78F 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 4

Loading and Configuration

4.1 MODULE Command

This command downloads and runs FF_RL78F module.

Command, form 1 (unload module)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	0xC0	20,40..43	

Command, form 2 (load module)

byte 0	byte 1	byte 2	
len	ecu	cmd	
N=10+m+n	0xC0	20,40..43	

byte 3	...	byte x	byte x	byte x	...	byte N-7	byte N-6	
mod 1	...	mod m	EOS	par 1	...	par n	EOS	
	...		0		...		0	

byte N-5	byte N-4	byte N-3	byte N-2	byte N-1	byte N
freq 1	freq 2	freq 3	freq 4	dummy	cks
				0	

len length of telegram
ecu target address
cmd command code

mod	filename of module (here: ff_rl78f.mod)
EOS	end-of-string of module filename (0)
par	name of parameter file, provided by CSM for the desired controller type, see remarks.
EOS	end-of-string of parameter file (0)
freq 1..4	These four bytes determine the clock frequency by which the controller is driven. It must be given in a special form: freq 1..3 is a mantissa with 3 digits and freq 4 is the exponent, one digit. Only the low nibbles of that bytes are used. All together it represents a number in the form of $XXX \cdot 10^Y$ which contains the clock frequency in Hz. Example (8MHz): 8 0 0 4
dummy	not used here, should be 0
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

- *par* is a parameter file which comes in binary format. It is NOT the ASCII parameter file provided by RENESAS but a special one which is adjusted by CSM for
 - the supported controller type
 - the used clock frequency

It is not easily substitutable for another controller even if it seems "nearly identical". Ask CSM when a new controller type should be supported.

- The freq entry is compared with a corresponding entry in the parfile by the FF_RL78F module. If it doesn't match, an error is reported by the module, and further operations are being blocked. That avoids programming the flash

with a wrong clock frequency assumption which may lead to a premature data loss of flash memory.

- After successful execution of the `MODULE` command, at least one of the interface slots should be configured for *MODULE interface* (15) using the `CONFIG_UNICOM(1)` command of the UCBASE software in order to use the commands implemented by FF_RL78F.
- After module startup, *CRC* computation is disabled, and the `CHECKSUM_FILE(76)` command (ref. chapter 6.10 on page 27) computes the RL78F specific checksum by adding all the involved bytes to a 16-bit-sum and computing the two's complement thereafter.
- If *par* is an empty string (consisting only of eos), FF_RL78F module is being started in *SCAN* mode, refer to chapter 3 on page 7.
- If *par* is a path name (must end with '\'), FF_RL78F module is being started in *AUTO* mode, refer to chapter 3 on page 7. The path must point to the directory 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 ".\".

4.2 CONFIG_INTERFACE Command

With this command the GPIOs for the signals RESET and TOOL0 and the type of file that contains the programming data (for FASTFLASH) can be selected.

Command, form 1

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

Command, form 2

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6
len	ecu	cmd	slot	config	filetype	cks
6	0xC0	4	0..3		0,4,8	

len	length of telegram
ecu	target address
slot	interface slot which leads to the FF_RL78F module
cmd	command code
config	configuration of GPIO setting for RESET and TOOL0 refer to remarks for more information
filetype	selects type of file that contains the programmin data: 0: BINARY, 4: SRECORD, 8: INTELHEX. Default: BINARY
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

- Coding of *config* parameter
The following configuration can be selected:
 - 0x00 (no bit set, standard configuration)
 - * GPIO7 (DSUB62 Pin49) for RESET
 - * GPIO8 (DSUB62 Pin50) for TOOL0
 - 0x01 (bit 0 = 1)
 - * GPIO1 (DSUB62 Pin7) for RESET
 - * GPIO5 (DSUB62 Pin29) for TOOL0
 - 0x02 (bit 1 = 1)
 - * GPIO2 (DSUB62 Pin8) for RESET
 - * GPIO6 (DSUB62 Pin30) for TOOL0
 - 0x04 (bit 2 = 1)
 - * GPIO3 (DSUB62 Pin51) for RESET
 - * GPIO7 (DSUB62 Pin49) for TOOL0
 - 0x08 (bit 3 = 1)
 - * GPIO4 (DSUB62 Pin28) for RESET
 - * GPIO8 (DSUB62 Pin50) for TOOL0

Bits 4..7 will be ignored. Do not specify more than one "1" bits with the *config* parameter!
- If *filetype* is set to none binary (4 or 8), the used file with the programming data must contain addresses in strongly ascending order without any gaps (one monolithic data block).
- Refer to the FASTFLASH section in *ucbase.pdf* to learn more about using hex files with the X_FASTFLASH(14) command.

Chapter 5

FASTFLASH

X_FASTFLASH(14) and X_FASTFLASH_MULTI(15) of the UCBASE software can be used with the FF_RL78F module. BINARY, SRECORD or INTEL_HEX files can be used. Refer to CONFIG_INTERFACE(4) command how to select (chapter 4.2 on page 11). Suppression of "0x00" or "0xFF" portions is not supported.

Refer also to `ucbase.pdf` for more information about these commands.

Chapter 6

FF_RL78F Commands

6.1 FF_RL78F::ENTER_PROGMode (1)

This command let target controller enter its programming mode. It must be executed before any other commands can be used that deals with the flash memory of target device. Additionally it controls the switching of the VOUTx voltage of UNICOM3 (VOUT1 (Pin 12 - DSUB62) and VOUT2 (Pin 10 DSUB62)).

Command (Form 1)

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

Command (Form 2)

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	
len	ecu	cmd	sw_voutx	baud_idx	supply	
9	xx	1	0..3	0..3	33..50	

byte 6	byte 7	byte 8	byte 9
config	dummy	dummy	cks
0,1	0	0	

len length of telegram
ecu target address
cmd command code

sw_voutx	switchin VOUTx: 0x00/0x01/0x02/0x03 0x00 -> VOUTx = off, 0x03 -> VOUTx = on, 0x01 -> VOUT1 = on, 0x02 -> VOUT2 = on
baud_idx	<i>Baud Rate Index</i> , 0: 115200, 1:250000, 2:500000, 3:1000000
supply	Supply voltage in units to tenth of volts (50 or 33)
config	bit 0 controls configuration of the <i>reset</i> pin: bit 0 = 0: reset pin is push-pull (default) bit 0 = 1: reset pin is open drain
dummy	not used, should be 0. for compatibility to the NEC_xxx modules of UNICOMII+
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			...		

Remarks

- This command must be executed before any flash relevant command can be used.
- If FF_RL78F module has been started in SCAN or AUTO mode, the command always reports the silicon signature of uC which was recognized. Refer to chapter 3 on page 7.

6.2 FF_RL78F::READ_VERSION (2)

This command reports about the module version information.

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 checksum of telegram

Remarks

- As version string `ff_rl78f_000Vx.yy` should be reported.

6.3 FF_RL78F::GENERIC_COMMAND (3)

With help of this command it is possible to send a simple command with parameters to the RL78F controller and fetchs its response.

Command

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte N-1	byte N
len	ecu	cmd	com	para 1	...	para n	cks
N=4+n	xx	3			...		

len	length of telegram
ecu	target address
cmd	command code
com	command code for the RL78F controller in programming mode
para X	additional parameters according to the command code
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	...	byte 4	byte 5
len	ecu	status	data 1	...	data n	cks
N=3+n	xx			...		

len	length of telegram
ecu	source address
status	result status
data X	response data of RL78F controller
cks	checksum of telegram

Remarks

- The ENTER_PROGMODE(1) command (ref. chapter 6.1 on page 14) must be executed successfully before this command can be used.

6.4 FF_RL78F::GET_LAST_STATE (4)

In case of failing of a command, this function makes it possible to retrieve the last returned status byte from the RL78F controller. The meaning of that byte must be figured out from the corresponding programming specification document.

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, state available

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	status	RL78F state	cks
4	xx			

Response, state not available

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

len length of telegram
ecu source address
status result status
RL78F state response status of RL78F controller
cks checksum of telegram

Remarks

- The ENTER_PROGMODE(1) command (ref. chapter 6.1 on page 14) must be executed successfully before this command can be used.

6.5 FF_RL78F::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 4096 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
N	xx	5	0/1	

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_RL78F 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.

6.6 FF_RL78F::CONFIG_CRC (6)

This command let the CHECKSUM_FILE(76) command (ref. chapter 6.10 on page 27) compute a CRC over the file instead of the RL78F specific checksum. The type of CRC is parametrizable in a wide range.

Command form 1, enabling CRC computation

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	
len	ecu	cmd	width	poly			
18	xx	6	1..32	MSB	...	LSB	

	byte 8	...	byte 11	byte 12	byte 13	
	init			refin	refout	
	MSB	...	LSB	0, 1	0, 1	

	byte 14	...	byte 17	byte 18
	xorout			cks
	MSB	...	LSB	

Command form 2, back to RL78F checksum

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
width	CRC width in bits, 1..32
poly	polynomial of CRC computation
init	initial value of CRC computation
refin	if not 0, input will be reflected
refout	if not 0, output will be reflected
xorout	value that is xored to the computation result at end
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 executing form 1 of command, CRC computation with the CHECK-SUM_FILE(76) command is enabled.

Following table shows some example CRC configurations:

CRC type	w'th	poly	init	r'n	r't	xorout
CSM CRC 16bit	16	0x00008005	0x0000CAC2	1	1	0x00000000
CSM CRC 32bit	32	0x04C11DB7	0xFFFFFFFF	1	1	0xFFFFFFFF
CRC CCITT 16bit	16	0x00001021	0x0000FFFF	0	0	0x00000000

6.7 FF_RL78F::GET_INFO (7)

This command reports about the internal clock frequency of RL78F and the active flash programming mode.

Command

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

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	clock	mode	cks
5	xx				

len	length of telegram
ecu	source address
status	result status
clock	clock frequency in MHz
mode	0: full speed mode, 1: wide voltage range mode
cks	checksum of telegram

Remarks

- The ENTER_PROGMODE(1) command (ref. chapter 6.1 on page 14) must be executed successfully before this command can be used.

6.8 FF_RL78F::GET_SIGNATURE (8)

This command reports the complete silicon signature of the RL78F.

Command

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

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 22	cks
25	xx			...		

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

6.9 FF_RL78F::CHECKSUM (74)

This command let the RL78F controller compute a checksum over its entire flash memory. That checksum is the two's complement of the 16 bit value which is the result of adding all bytes of the flash memory range.

Command

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	opt	
12	xx	74	0	

	byte 4	byte 5	byte 6	byte 7	
	start				
	MSB			LSB	

	byte 8	byte 9	byte 10	byte 11	byte 12
	end				cks
	MSB			LSB	

len	length of telegram
ecu	target address
cmd	command code
opt	not used here, should be 0
start/end	These values determine the start and end address of checksum computation. The start address must match with the start address of a flash block/area, and the end address must match with an end address of a flash block/area. Checksum computation may be done over more then one block/area or over the whole flash memory. In this case, start address must be the start address of flash memory and end address must be the end address of flash memory.
cks	checksum of telegram

Response

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

len	length of telegram
ecu	source address

status	result status
checksum	checksum computation result
cks	checksum of telegram

Remarks

- The ENTER_PROGMODE(1) command (ref. chapter 6.1 on page 14) must be executed successfully before this command can be used.

6.10 FF_RL78F::CHECKSUM_FILE (76)

That command computes a checksum over a range of a given file with the same algorithm as RL78F controller it does (ref. chapter 6.9 on page 25), or a CRC which type is defined with the CONFIG_CRC(6) command (ref. chapter 6.6 on page 21). If merging data are defined and enabled (s. SET_MERGE(5), ref. chapter 6.5 on page 19), 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

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	not used here, should be 0
start/end	start and end address as in CHECKSUM(74) command
offset	offset in file
file	(optional) name of file that contains programming data, same as used for FASTFLASH (BINARY file only!)
EOS	(optional) end-of-string (0), must be inserted if <i>file</i> is specified
cks	checksum of telegram

Response

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

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

Remarks

- The command can only be executed if BINARY file is selected. See CONFIG_INTERFACE(4), chapter 4.2 on page 11).
- That command should be used in order to compute the expected checksum over the file with applied merge data. After programming via FASTFLASH, the CHECKSUM (74) command should report the same checksum as calculated with this command.
- If no *file* is specified, DEFAULT.DAT is assumed.
- If CRC computation is enabled, the result could be merged into the flash image file using the SET_MERGE(5) command (ref. chapter 6.5 on page 19) into the right place where the application software of RL78F expects it.

6.11 FF_RL78F::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 6.5 on page 19).

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.

6.12 FF_RL78F::SECURITY (110)

This command can read, reset or define security settings of the RL78F controller.

Command (form 1, read or reset security settings)

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	opt	cks
4	xx	110	0,1	

Command (form 2, set security settings)

byte 0	byte 1	byte 2	byte 3	byte 4	
len	ecu	cmd	flg	bot	
9	xx	110			

	byte 5	byte 6	byte 7	byte 8	byte 9
	ss		se		cks
	MSB	LSB	MSB	LSB	

len	length of telegram
ecu	target address
cmd	command code
opt	(form 1 only) 0: read settings, 1: reset settings
flg	security flags
bot	Boot block cluster number
ss	Flash shield window start block number
se	Flash shield window end block number
cks	checksum of telegram

Response to form 2 and form 1 reset security settings)

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 to form 1 read security settings

byte 0	byte 1	byte 2	byte 3	byte 4	
len	ecu	status	flg	bot	
9	xx				

	byte 5	byte 6	byte 7	byte 8	byte 9
	ss		se		cks
	MSB	LSB	MSB	LSB	

Remarks

- The ENTER_PROGMODE(1) command (ref. chapter 6.1 on page 14) must be executed successfully before this command can be used.
- if security options should be reset, a *ERASE_FLASH(115)* command must be issued before (s. chapter 6.13 on page 33).

6.13 FF_RL78F::ERASE_FLASH (115)

This command erases either the entire flash memory of RL78F (chip erase) or a part of it (block erase).

Command, Chip Erase

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

Command, Block Erase

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

byte 4	byte 5	byte 6	byte 7	
start				
MSB			LSB	

byte 8	byte 9	byte 10	byte 11	byte 12
end				cks
MSB			LSB	

len	length of telegram
ecu	target address
cmd	command code
opt	not used here, should be 0
start/end	erase address range. All the flash blocks that are touched by it, are being erased.
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 ENTER_PROGMODE(1) command (ref. chapter 6.1 on page 14) must be executed successfully before this command can be used.
- This command should be executed before FASTFLASH, because flash memory should be in erased state before programming.

6.14 FF_RL78F::SWITCH_VERIFY (117)

This command controls whether FASTFLASH programs data from file into flash of uC, or does a verify operation.

Command

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	mode	cks
4	xx	117	0,1	

len	length of telegram
ecu	target address
cmd	command code
mode	0: program mode, 1: verify mode. default after MODULE(20): program 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

- This type of verify operation is intended to use together with none binary program files since there it is often difficult to generate a checksum which could be compared using the CHECKSUM(74) command (ref. chapter 6.9 on page 25). It is much slower then CHECKSUM command since data must be transferred twice to the target device. It is suggested always to use BINARY files where the checksum can simply be computed by an external tool or by CHECKSUM_FILE(76) command (ref. chapter 6.10 on page 27).

6.15 FF_RL78F::CHECK_FLASH (124)

This command performs a blank check over the entire flash memory or over a part of it.

Command form 1, complete check

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

Command form 2, address range based check

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

byte 4	byte 5	byte 6	byte 7	
start				
MSB			LSB	

byte 8	byte 9	byte 10	byte 11	byte 12	byte13
end				d01	cks
MSB			LSB	0,1	

len	length of telegram
ecu	target address
cmd	command code
opt	not used here, should be 0
start/end	address range in which the blank check is to performed
d01	check of security bits is being included. If at least one of these bits is set to 0, check failes with BLANK_CHECK_ERROR.
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 ENTER_PROGMODE(1) command (ref. chapter 6.1 on page 14) must be executed successfully before this command can be used.
- The command should be executed after ERASE_FLASH(115) (ref. chapter 6.13 on page 33) in order to ensure that flash memory is really blank.

6.16 FF_RL78F::ErrorCodes

The following table describes possible error codes reported by the *status* of the response telegrams, and their meanings. Only the FF_RL78F specific error codes are described here. Furthermore, the error codes described in `ucbase.pdf` can occur.

Error	Code	Description
NO_ERROR	0xA0	No error occurred
RL78_TIMEOUT_ERROR	0xD0	RL78F controlled didn't respond within timeout time
RL78_FORMAT_ERROR	0xD1	bad frame received from RL78F controller
RL78_BUFFER_OVERFLOW	0xD2	response from RL78F controller larger as expected
RL78_CHECKSUM_ERROR	0xD3	bad frame checksum detected
RL78_STATUS_ERROR	0xD4	unexpected state of RL78F controller received
RL78_PROGMODE_FAILED_ERROR	0xD5	programming mode not reached
RL78_ERASE_ERROR	0xD6	flash erase failed
RL78_PROGRAM_ERROR	0xD7	flash programming failed
RL78_BLANKCHECK_ERROR	0xD8	blank check processing failed
RL78_VERIFY_ERROR	0xD9	error while verifying the flash memory
RL78_MERGE_ERROR	0xDF	adding of merge area failed, no space
RL78_COMMAND_ERROR	0xE0	RL78F reported bad command
RL78_PARAMFILE_ERROR	0xE1	parameter file doesn't match with module version
RL78_CLOCK_FREQ_MISMATCH	0xE2	clock frequency in parameter file and in module command differs
RL78_NOT_IN_PROGMODE_ERROR	0xE3	command could not be executed since target uC is not in programming mode
RL78_WRONG_SILICON_SIGNATURE	0xE4	silicon signature doesn't match with the one in parameter file
RL78_IN_SCAN_MODE_ERROR	0xE5	command could not be executed since target uC is in scan mode
RL78_UNSUPP_FREQUENCY_ERROR	0xF6	clock frequency specified which is not supported by FF_RL78F
RL78_SUPPLY_ERROR	0xEF	no supply voltage at V_GPIO pin detected