

SWD_ARM Module Documentation

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

July 8, 2025

Date	Version	Name	Changes
2019-01-17	1.0	CSM/RN	first release
2020-02-12	1.1	CSM/RN	S32K support
2020-04-03	1.3	CSM/RN	TR2 support
2020-04-20	1.4	CSM/RN	more error codes
2020-05-04	1.5	CSM/SC	uC selector for TRV2_CM0P added
2021-01-25	1.6	CSM/SC	uC selector for TRV2_CM0P only added
2021-03-02	1.7	CSM/SC	SWD IDCODE for MODULE(20,...) response added
2021-06-16	1.8	CSM/RN	GPIO routing
2021-09-17	2.0	CSM/RN	UBATT control
2021-10-27	2.1	CSM/RN	NCJ29 support
2022-09-21	2.5	CSM/RN	ATIC543 support
2023-03-09	2.6	CSM/RN	UNICOM4 support, PIC32CM support
2023-03-16	2.8	CSM/RN	PIC32CM unsecure
2023-07-04	3.2	CSM/RN	MERGE, CRC commands
2023-08-03	3.3	CSM/RN	verify via block check crc
2023-11-30	3.4	CSM/RN	S9KEA support
2023-12-18	3.5	CSM/SC	TLE987x support
2024-01-22	3.6	CSM/RN	alternate start address
2024-03-19	3.7	CSM/SC	HVC5x MCU type added
2024-08-21	3.8	CSM/RN	MSPM0 MCU type added
2024-09-20	3.9	CSM/RN	More error codes
2024-11-04	4.0	CSM/RN	ADJUST_BITRATE command code fixed
2024-11-06	4.1	CSM/RN	READ_MEMORY command fixed
2025-03-10	5.0	CSM/RN	Option IO support, Sampling Point adjustment
2025-04-04	5.1	CSM/RN	Debug commands
2025-07-08	5.2	CSM/RN	uC list 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 © 2025 by CSM GmbH

Contents

1	Introduction	3
2	Overview	4
3	Loading and Configuration	6
3.1	MODULE Command	6
3.2	CONFIG_INTERFACE Command	10
4	FASTFLASH	11
5	SWD_ARM Commands	12
5.1	SWD_ARM::ADJUST_BITRATE (1)	12
5.2	SWD_ARM::READ_VERSION (2)	13
5.3	SWD_ARM::SET_MERGE (5)	14
5.4	SWD_ARM::CRC_INIT (75)	16
5.5	SWD_ARM::CRC_FILE (76)	18
5.6	SWD_ARM::GEN_IMAGE (77)	20
5.7	SWD_ARM::CRC_DATA (78)	22
5.8	SWD_ARM::SET_GET_REGISTER (100)	23
5.9	SWD_ARM::HALT_CORE (101)	25
5.10	SWD_ARM::WAIT_HALT (102)	26
5.11	SWD_ARM::SET_BREAKPOINT (103)	27
5.12	SWD_ARM::SINGE_STEP (104)	28
5.13	SWD_ARM::WRITE_MEMORY (112)	29
5.14	SWD_ARM::READ_MEMORY (113)	30
5.15	SWD_ARM::SWITCH_VERIFY (117)	31
5.16	SWD_ARM::CORE_RESET (127)	32
5.17	SWD_ARM::ErrorCodes	33

Chapter 1

Introduction

SWD_ARM is a module for extending the *UCBASE* software running on *UNI-COM*.

It implements communication with ARM Cortex based target devices thru its *Serial Wire Debug (SWD)* interface such as downloading toolbox code, communication with running toolbox, performing high speed flash programming or simply writing and reading memory.

Chapter 2

Overview

To use UNICOM device with SWD_ARM module, UCBASE software version 3.Ax (UNICOM3 Rev.C), version 3.Cx (UNICOM3 Rev.C1), version 4.Ax (UNICOM3 Rev.D) resp. version 5.1x (UNICOM4) or newer must be installed on UNICOM.

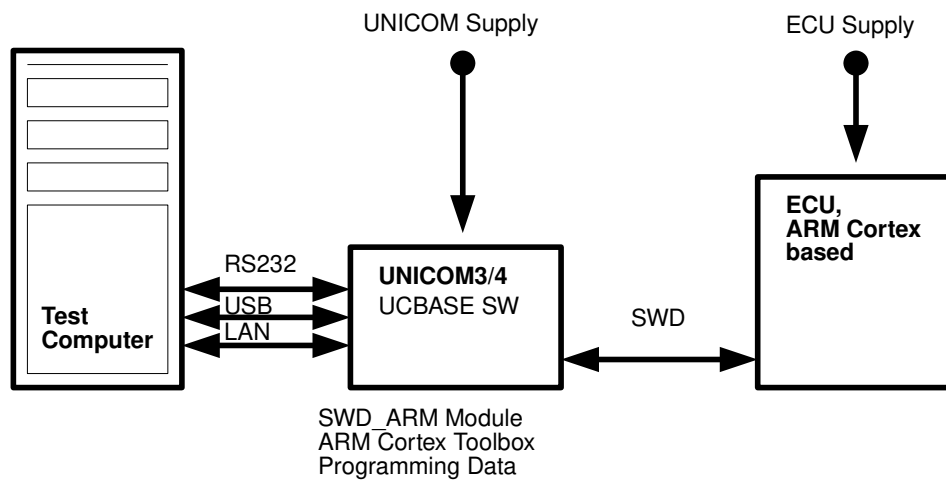


Figure 2.1: Components of the System

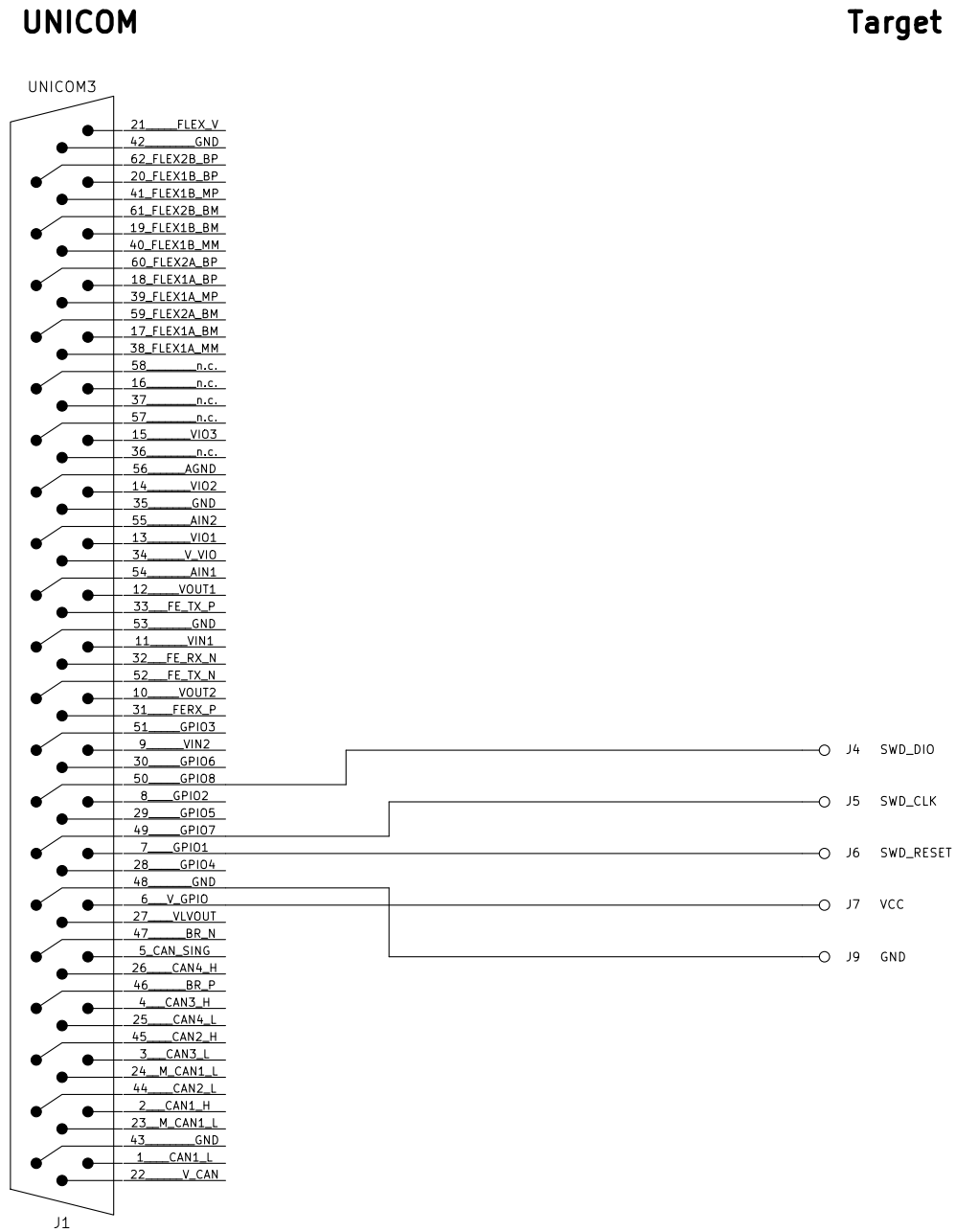


Figure 2.2: Connection Diagram

Chapter 3

Loading and Configuration

3.1 MODULE Command

This command runs the SWD_ARM module. Furthermore, the operating system can be downloaded to the ECU.

Command, form 1 (unload module)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	0xC0	20,40..42	

Command, form 2 (load module)

byte 0	byte 1	byte 2	byte 3	...	byte m+2	byte m+3	
len	ecu	cmd	mod 1	...	mod m	EOS mod	
N	0xC0	20,40..42				0	

	byte m+4	...	byte x	byte x	byte x	byte x	byte x	byte x	
	tbx 1	...	tbx k	EOS tbx	tbx addr				
				0	MSB			LSB	

	byte x	byte x	byte x	byte x	byte x	byte x	
	SSC	CONFIG	CONFIG2	RES	CLK	DIO	
				1..11	1..8	1..8	

	byte x	byte x	byte x	byte x	
	UB_ON_DEL		UB_OFF_DEL		
	MSB	LSB	MSB	LSB	

	byte x	...	byte x	byte x	byte x	byte x	byte x	byte x	
	pw 0	...	pw n	start addr				sp	
		...		MSB			LSB	0..7	

	byte x	...	byte N-1	byte N
	params0	...	params31	cks
		...		

len	length of telegram
ecu	target address
cmd	command code
mod	filename of module
EOS mod	end-of-string of module filename (0)
tbx	(optional) filename of OS or monolythic toolbox. if length is 0, no OS/toolbox will be loaded by the module
EOS mod	end-of-string of OS filename (0), must exists even with length of <i>tbx</i> is 0.
tbx addr	destination address of code in RAM. If 0xFFFFFFFF is specified, AUTO mode is selected (works not with all OSes). Without <i>tbx</i> it must exist but it is dummy.
SSC	SWD bitrate in units to 100 kBits/s (max. 250 with UNICOM4 Rev.E and 50 alse).
CONFIG	Control flags: <ul style="list-style-type: none"> bit 7: 1: CONFIG2 follows bit 6: 1: three GPIO routing bytes follow: RES, CLK, DIO bit 5: 1: An unsecure procedure is performed bit 4: 1: Target RESET is controlled via PUSH/PULL, O-PENDRAIN else bits 0..3: code for uC: <ul style="list-style-type: none"> 0: default 1: MKW3x 2: S32K 3: Traveo TRV2_CM4 (Cortex-M4) 4: Traveo TRV2_CM0P (Cortex-M0+) 5: Traveo TRV2_CM0P (Cortex-M0+ Core only) 6: NCJ29D5 7: ATIC543 8: PIC32CM

	9: S9KEA
	10: TLE987x
	11: HVC5x
	12: MSPM0
CONFIG2	(optional, depending on <i>CONFIG:7</i>), more control flags
	bit 0: 1: UBatt controll parameters follow
	bit 1: 1: uC specific password follows
	bit 2: 1: (UNICOM4 Rev.E only) Enables LVDS lines (Option.IO Adapter) instead of GPIOs
	bit 4: 1: (UNICOM3 Rev.D and UNICOM4 Rev.E only) Sample Point Offset follows
RES	(optional, depending on <i>CONFIG:6</i>) GPIOx or VIOx which is used as SWD RESET line (default: 1)
CLK	(optional, depending on <i>CONFIG:6</i>) GPIOx which is used as SWD CLK line (default: 7)
DIO	(optional, depending on <i>CONFIG:6</i>) GPIOx which is used as SWD DIO line (default: 8)
UB_ON_DEL	(optional, depending on <i>CONFIG2:0</i>) defines a delay after switching power on thru VIN1/VOUT2 by the module in milliseconds)
UB_OFF_DEL	(optional, depending on <i>CONFIG2:0</i>) defines a delay after switching power off thru VIN1/VOUT2 by the module in milliseconds)
pw x	(optional, depending on <i>CONFIG2:1</i>) defines a uC specific password for unlocking. Length of password depends on uC. Not supported for all uCs.
start addr	(optional, depending on <i>CONFIG2:3</i>) defines an alternate start address for downloaded code
sp	Sample Point offset, 0..7, in steps of 10 ns (UNICOM4 Rev.E) resp. 12.5 ns (UNICOM3 Rev.D) If 0, sampling happens exactly on the natural sampling slope (the opposite one of which is used for data shifting). As greater this value, as later the sampling, it may exceed the end of current bit. Default value is 6. This can be used to compensate signal delays on cable and target devices.
params	(optional) additional parameters that are forwarded to the OS
cks	checksum of telegram

Response

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

byte 0	byte 1	byte 2	byte 3	...	byte 6	byte 7
len	ecu	status	idcode			cks
7	0xC0	0xA0	MSB	...	LSB	

len	length of telegram
ecu	source address
status	result status
idcode	SWD IDCODE
cks	checksum of telegram

Remarks

- After loading the module, at least one interface slot must be configured for *MODULE* interface using the *CONFIG_UNICOM(1)* command of UCBASE software.
- The SWD_ARM module supports forwarding of STP commands to the OS thru the SWD interface. To distinguish between commands for SWD_ARM itself and such for OS, the *ECU* number must be set accordingly. If, for instance, slot 0 is configured for *MODULE* interface and ECU number of command from test computer is 0x80, the command will be interpreted by SWD_ARM itself. If ECU number is 0x00, the command is forwarded to the OS.

Refer to *ucbase.pdf* to learn more about using ECU numbers in command telegrams.

- The size of the parameter area is restricted to 32 bytes.
- If GPIO routing is used, all the parameters RES, CLK and DIO must have different values and they must fit their range. RES can be 1..11. 1..8 corresponds to GPIO1..8, 9..11 corresponds to VIO1..VIO3. CLK and DIO can be 1..8 (GPIO1..8). On UNICOM3 Rev.C, DIO is limited to GPIO5..8 due to hardware considerations. If VIOx should be used as RESET line, V_VIO (DSUB34) must be connect to VCC of target or V_GPIO (DSUB6).
- With S9KEA uC, flash programming only works if no application software is inside the flash memory. If flash is already programmed, execute *unsecure procedure* first!

3.2 CONFIG_INTERFACE Command

This command is normally used for configuring parameters of the interface which is activated on the specified slot. In case of MODULE interface is active, the command can usually configure the module which is reachable over this slot.

With SWD_ARM module, the command adjusts type of file which contains the programming data and which is used by FASTFLASH.

Command

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

len	length of telegram
ecu	target address
cmd	command code
slot	interface slot over which the SWD_ARM module can be reached
par	file type selector: 0: BINARY type (default after module start) 4: SRECORD type 8: INTEL HEX type else: forces PARAMETER_ERROR
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

Chapter 4

FASTFLASH

SWD_ARM realizes *FASTFLASH* thru SWD interface with help of running tool-box on target side.

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

The type of file that contains the programming data can be selected by the *CONFIG_INTERFACE(4)* Command, refer chapter 3.2 on page 10.

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

FASTFLASH can also be used for verifying programmed data. Refer to *SWITCH_VERIFY(117)* command (chapter 5.15 on page 31).

Chapter 5

SWD_ARM Commands

5.1 SWD_ARM::ADJUST_BITRATE (1)

This command can be used to select a new SWD bitrate.

Command

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

len	length of telegram
ecu	target address
cmd	command code
bitr	the new SWD bitrate in units of 100kBits/s.
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

5.2 SWD_ARM::READ_VERSION (2)

This command reports about the version information of SWD_ARM.

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	check
19	xx					

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

Remarks

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

5.3 SWD_ARM::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 blocks of merging data and a total data amount of 4096 bytes is 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	0/1/2	

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 SWD_ARM 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.

5.4 SWD_ARM::CRC_INIT (75)

This command selects the CRC computation algorithm for CRC_FILE (ref. chapter 5.5 on page 18) and CRC_DATA (ref. chapter 5.7 on page 22).

Command form 1, select a pre-defined algorithm

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	index	cks
4	xx	75		

Command form 2, define parameters of algorithm

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7
len	ecu	cmd	width	poly		
18	xx	75	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	

len	length of telegram
ecu	target address
cmd	command code
index	index to one of the pre-defined algorithms: 0: CSM CRC16 (default after start of module) 1: CSM CRC32 = XCP CRC32 = ZIP CRC 2: RH850 CRC32 3: XCP CRC16 4: CCITT CRC16
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

Following table shows some example CRC configurations for form 2 (which also can be selected by form 1 via index):

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

5.5 SWD_ARM::CRC_FILE (76)

That command computes a configurable CRC over a range of a given file.

If merging data are defined and enabled (s. SET_MERGE(5), ref. chapter 5.3 on page 14), that data is taken into account for the crc 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
offset	offset in file
file	name of file that contains programming data, same as used for FASTFLASH (BINARY file only!)
EOS	end-of-string (0)
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	...	byte N-1	byte N
len	ecu	status	crc		cks	
N	xx		MSB	...	LSB	

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

Remarks

- The command can only be executed if BINARY file is selected. See MODULE, *options* parameter, chapter 3.1 on page 6).
- 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 check over the flash memory should report the same CRC as computed over the file.
- The CRC computation algorithm can be selected and adjusted by using the CRC_INIT(6) command (ref. chapter 5.4 on page 16). By default, after loading the module, CSM CRC16 is selected.

5.6 SWD_ARM::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 5.3 on page 14).

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.

5.7 SWD_ARM::CRC_DATA (78)

The command computes a configurable CRC over data bytes contained by this command.

Command

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte N-1	byte N
len	ecu	cmd	opt	data	...	data	cks
N	xx	78	0		...		

len	length of telegram
ecu	target address
cmd	command code
data	data bytes which are involved into CRC computation
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3	...	byte N-1	byte N
len	ecu	status	crc			cks
N	xx		MSB	...	LSB	

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

Remarks

- The CRC computation algorithm can be selected and adjusted by using the CRC_INIT(6) command (ref. chapter 5.4 on page 16). By default, after loading the module, CSM CRC16 is selected.

5.8 SWD_ARM::SET_GET_REGISTER (100)

With this command, general purpose or status registers of ARM core can be set or read.

Command, read form

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

Command, set form

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	byte 8
len	ecu	cmd	reg	val			cks
8	xx	100		MSB	...	LSB	

len	length of telegram
ecu	target address
cmd	command code
reg	register code: 0..12: R0..R12 13: current SP 14: LR 15: Debug Return Address (= PC) 16: xPSR / Flags, Execution Number, and state information 17: Main SP 18: Process SP 20: CONTROL bits [31:24], FAULTMASK bits [23:16], BASEPRI bits [15:8], and PRIMASK bits [7:0]
val	new value (for set form)
cks	checksum of telegram

Response, read form

byte 0	byte 1	byte 2	byte 3	...	byte 6	byte 7
len	ecu	cmd	val			cks
7	xx	100	MSB	...	LSB	

Response, set form

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

len	length of telegram
ecu	source address
status	result status
val	value of register
cks	checksum of telegram

Remarks

- Command can only be executed if core is halted (refer HALT_CORE, chapter 5.9 on page 25).

5.9 SWD_ARM::HALT_CORE (101)

The command can be used to halt or restart ARM core.

Command

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	halt	cks
4	xx	101		

len length of telegram
ecu target address
cmd command code
halt 1: halt, 0: restart
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 MODULE(20) command without specified OS/toolbox, core is already halted.
- After MODULE(20) command with specified OS/toolbox, core runs and executes OS/toolbox code.
- If general purpose registers of core should be read or set (ref. chapter 5.8 on page 23), core must be in halt state.
- If core is halted by reaching a break point, this break point must be deleted before the core can be restarted again (ref. chapter 5.11 on page 27).

5.10 SWD_ARM::WAIT_HALT (102)

The command waits until core has been halted.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	102	

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

- Timeout time is a fixed value of 1 second.

5.11 SWD_ARM::SET_BREAKPOINT (103)

with this command, break points can be set or cleared

Command, form1: clear all breakpoints

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	103	

Command, form2: clear one breakpoint

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	comp	cks
4	xx	103		

Command, form3: set one breakpoint

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	byte 8
len	ecu	cmd	comp	addr			cks
8	xx	103		MSB	...	LSB	

len	length of telegram
ecu	target address
cmd	command code
comp	number of compare unit
addr	address of breakpoint
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

- Maximum number of compare unit depends on core type. A minimum of 4 units should always be available.

5.12 SWD_ARM::SINGE_STEP (104)

The command let core execute one instruction and halts it again.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	104	

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

Response

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

len length of telegram
ecu source address
status result status
PC value of program counter after execution of instruction
cks checksum of telegram

Remarks

- Core must be in halted state before the command can be executed
- If core is halted by reaching a break point, this break point must be deleted before the core can do a single step (ref. chapter 5.11 on page 27).

5.13 SWD_ARM::WRITE_MEMORY (112)

With this command, data can be written to any address of ARM Cortex.

Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7	
len	ecu	cmd	opt	addr				
N=8+n	xx	112	1,2,4	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	Type of data to be written: 1: byte(8bit), 2: word(16bit), 4: dword(32bit)
addr	address to which data is written
data	data bytes that should be written
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

5.14 SWD_ARM::READ_MEMORY (113)

With this command, data can be read from any address of ARM Cortex.

Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7	
len	ecu	cmd	opt	addr				
10	xx	113	1,2,4	MSB			LSB	

byte 8	byte 9	byte 10
size		cks
MSB	LSB	

len	length of telegram
ecu	target address
cmd	command code
opt	Type of data to be written: 1: byte(8bit), 2: word(16bit), 4: dword(32bit)
addr	address from which data is read
size	amount of bytes that should be read, starting from the given address
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	check
N=3+n	xx					

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

5.15 SWD_ARM::SWITCH_VERIFY (117)

This command selects the behaviour of *X_FASTFLASH(14)* command, either *programming mode* (default setting) or *verify mode*.

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: programming 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

- In *program mode*, FASTFLASH transfers the programming data portion by portion to the target device and let it be programmed into the flash memory.
- In *verify mode*, FASTFLASH will not transfer any data. It analyzes the file for contiguous data blocks, computes the CRC over them and let the target device do the same on the corresponding flash areas.
- This method is the suggested only if the flash data are contained by an SRECORD or INTEL HEX file where no CRC can be computed easily, or merged data are being active (ref. chapter 5.3 on page 14).

5.16 SWD_ARM::CORE_RESET (127)

The command resets ARM core in different ways.

Command

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	opt	cks
4	xx	127		

len	length of telegram
ecu	target address
cmd	command code
opt	0: Power On Reset (Power Cycle) 1: External Reset (Pulse on RESET line) 2: Debug Reset (Software Reset via AIRCR Debug Register)
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

- With opt=0 or 1, the command only works if target device is wired accordingly.

5.17 SWD_ARM::ErrorCodes

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

Error	Code	Description
NOT_INITIALIZED_ERROR	0x90	Not initialized
INVALID_OPTION_ERROR	0x9F	Invalid option parameter specified
NO_ERROR	0xA0	No error occurred
PARAMETER_ERROR	0xB0	Wrong parameter in command telegram
LENGTH_ERROR	0xB3	Wrong command telegram length
ADDRESS_ERROR	0xB7	Invalid address parameter in command telegram
FILE_ERROR	0xB9	File could not be opened, or it is corrupted
TEL_TOO_LONG_ERROR	0xB8	Command telegram too long for forwarding to ECU
ECU_CHECKSUM_ERROR	0xC2	Telegram received from ECU with wrong checksum
ECU_LENGTH_ERROR	0xC3	Response telegram from target is too long
ECU_RECEIVE_ERROR	0xC4	Invalid SWD ACK response received from target (wiring problem?)
ECU_TIMEOUT_ERROR	0xC5	No response telegram received within specified timeout time
NO_VGPIO_ERROR	0xE0	VGPIO needs to be turned on
SWD_TRANSFER_ERROR	0xE1	Data transfer over SWD failed
SWD_PARITY_ERROR	0xE2	Parity error while receiving from SWD
TOOLBOX_FILE_ERROR	0xE3	Something wrong with the toolbox file
VERIFY_ERROR	0xE4	Error during verification while toolbox download
TOOLBOX_VERSION_ERROR	0xE5	Os/toolbox api version doesn't fit with the module
MASS_ERASE_DISABLED_ERROR	0xE6	Unsecure by mass erase is disabled
WRONG_ID_ERROR	0xE7	Wrong SWD ID received
PW_NOT_SUPPORTED_ERROR	0xE8	Password not supported with selected uC
PW_INVALID_ERROR	0xE9	Invalid password specified
WRONG_RESPONSE_ERROR	0xEA	wrong response from target while verify process
BLANK_CHECK_ERROR	0xEB	Flash not blank
NO_TRIM_VAL_AVAIL_ERROR	0xEC	No factory trim value found (S9KEA)
MERGE_ERROR	0xED	Error while defining merge areas
ENTER_DEBUG_ERROR	0xEE	Error when entering debug mode
EXIT_DEBUG_ERROR	0xEF	Error when exiting debug mode
UNKNOWN_COMMAND_ERROR	0xFF	Command code not supported by the module

More error codes are described in `ucbase.pdf`