

TC_DAP Module Documentation

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

July 9, 2024

Date	Version	Name	Changes
2015-03-13	1.0	CSM/RN	first release
2017-03-24	1.1	CSM/RN	for UNICOM3
2017-10-13	1.2	CSM/RN	download command
2018-05-25	1.3	CSM/RN	CONFIG_MODULE for UNICOM3 added
2020-01-14	1.4	CSM/SC	TC_TYPE command and ext. device params for UNICOM3 added
2020-04-14	1.5	CSM/RN	flash programming via DAP only
2020-10-27	1.6	CSM/SC	default TC_TYPE specified by OS
2021-04-13	1.7	CSM/RN	STP over DAP gateway and FAST-FLASH
2022-03-03	1.8	CSM/RN	Adjustable sample point
2022-12-19	1.9	CSM/RN	GO command, options with MODULE command
2023-03-03	2.0	CSM/TO	NO_VGPIO_ERROR description corrected
2023-10-20	2.1	CSM/RN	UNICOM4 support
2024-07-09	2.2	CSM/RN	Delayed Reset with DAP entry

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 © 2024 by CSM GmbH

Contents

1	Introduction	3
2	Overview	4
3	STP and FASTFLASH over DAP	6
4	Loading and Configuration	8
4.1	MODULE(20) command for TC_DAP	8
4.2	CONFIG_MODULE Command	11
5	FASTFLASH	13
6	TC_DAP Module Commands	14
6.1	TC_DAP::READ_VERSION (2)	15
6.2	TC_DAP::GET_LAST_STATE (3)	16
6.3	TC_DAP::DOWNLOAD (22)	17
6.4	TC_DAP::WRITE_REGISTER (102)	19
6.5	TC_DAP::READ_REGISTER (103)	20
6.6	TC_DAP::WRITE_MEM (112)	21
6.7	TC_DAP::READ_MEM (113)	22
6.8	TC_DAP::ERASE_CHECK_FLASH (115)	23
6.9	TC_DAP::PROGRAM_FLASH (116)	24
6.10	TC_DAP::SWITCH_VERIFY (117)	25
6.11	TC_DAP::READ_FLASH (123)	26
6.12	TC_DAP::GO (126)	27
6.13	TC_DAP::TC_TYPE (250)	28
6.14	TC_DAP::ErrorCodes	30

Chapter 1

Introduction

The *TC_DAP* FFCOMBI/UCBASE module realizes communication with *TriCore TC2xx and TC3xx (Aurix)* over *DAP* interface. It can be used for erasing and programming the flash memory in order to re-enable the bootstrap loaders after programming a software that locks it via *Boot Mode Index*, or restore content of UCB sectors. Further it is possible to read/write registers and RAM for test purpose.

An *STP over DAP* gateway is implemented that allows STP communication only over the DAP interface, also a FASTFLASH which uses only DAP.

Chapter 2

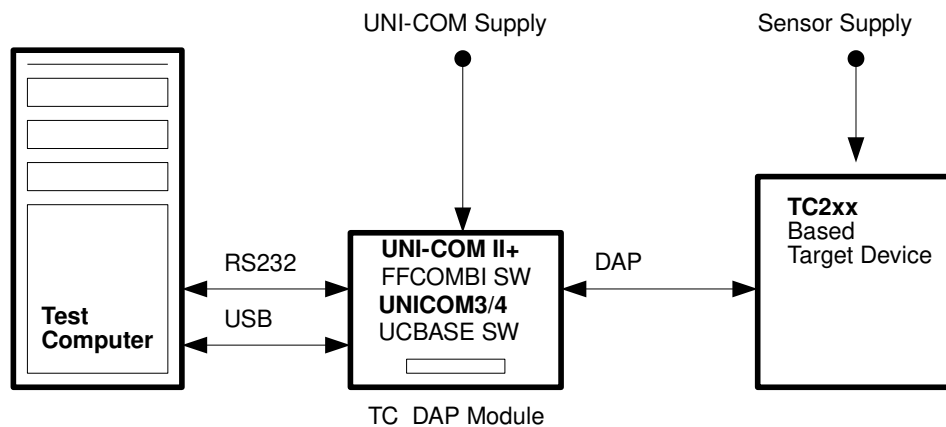
Overview

To use UNICOM II+ device with TC_DAP module, it must have at least a daughter board "ECU Signals". The recommended device type is "CSM UNI-COM II+ CombiFlasher" (art.no. 330649).

On UNICOM II+, the FFCOMBI software V9.x or newer must be installed.

The UNICOM3/4 variant of the module can be used with UNICOM3 revisions C, C1 and D and UNICOM4.

The figure below shows the components of the system.



For UNICOM II+, there is a small circuit necessary to connect the Aurix based target device to UNICOM. Please refer to the [connection.pdf](#) of the related software delivery package for more information.

For UNICOM3/4,

- *PORST* must be connected to GPIO2 (DSUB62 Pin 8)
- *DAP0* must be connected to GPIO4 (DSUB62 Pin 28)
- *DAP1* must be connected to GPIO8 (DSUB62 Pin 50)
- *VCC* must be connected to V_GPIO (DSUB62 Pin 6)
- *GND* must be connected to GND (e.g. DSUB62 Pin 43)
- A pulldown resistor of app. 1k0 must be connected between DAP1 and GND

The target device must be supplied so that VCC comes from device.

Chapter 3

STP and FASTFLASH over DAP

The TC_DAP module also allows STP and FASTFLASH communication by using the DAP interface only. That can be useful if no other interface (CAN, FlexRay...) is available on target device for fast data transfer.

Note that this type of communication is much slower than using CAN(FD) or FlexRay. Transfer speed is comparable with that one of a single CAN (1 MBits/s) without FD extension.

Following conditions must be met in order STP over DAP can be used:

- A OS/toolbox must be downloaded to the target device which is STP on DAP capable.
- At least one interface slot must be set to MODULE(15).
- STP commands, that should be redirected to target device over DAP must have an ECU number with bit 7 = 0, e.g. `0x03 0x00 0x02 <cks>` will be redirected to target and let it send its version information. In contrast `0x03 0x80 0x02 <cks>` will report the version information of TC_DAP module.

If an STP over DAP capable OS is loaded, TC_DAP uses a FASTFLASH protocol that also communicates with STP over DAP. This is much faster than programming the flash of target directly without OS/toolbox. If *Verify Mode* is active, FASTFLASH checks the flash content by computing a cumulative CRC over flash and file to the same time. That avoid reading back the flash data from target device for verification.

So there are now 2 ways to program flash memory directly over DAP:

- Using the flash commands inc. FASTFLASH of the TC_DAP module without loading any OS/toolbox. That is very slow.

- Using the commands of a loaded STP over DAP capable OS/toolbox for programming, that is much faster.

However, this should only be used if no other (faster) second interface for mass data transfer is available.

Chapter 4

Loading and Configuration

4.1 MODULE(20) command for TC_DAP

This command downloads and runs the TC_DAP 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	byte 3	...	byte N-2	byte N-1	byte N
len	ecu	cmd	mod 1	...	mod m	EOS	cks
N=4+m	0xC0	20,40..43				0	

Command, form 3 (load module, with option, UNICOM3/4 only)

byte 0	byte 1	byte 2	
len	ecu	cmd	
N=5+m	0xC0	20,40..43	

	byte 3	...	byte N-3	byte N-2	byte N-1	byte N
	mod 1	...	mod m	EOS	opt	cks
				0		

Command, form 4 (load module, unlock key)

byte 0	byte 1	byte 2	byte 3	...	byte N-34	byte N-33	
len	ecu	cmd	mod 1	...	mod m	EOS	
N=36+m	0xC0	20,40..43				0	

byte N-32	...	byte N-1	byte N
key 1	...	key 32	
	...		

Command, form 5 (load module, with option and unlock key, UNICOM3/4 only)

byte 0	byte 1	byte 2	byte 3	...	byte N-35	byte N-34	
len	ecu	cmd	mod 1	...	mod m	EOS	
N=37+m	0xC0	20,40..43				0	

byte N-33	byte N-32	...	byte N-1	byte N
opt	key 1	...	key 32	
		...		

len	length of telegram
ecu	target address
cmd	command code
mod	filename of module (here: TC_DAP), without .C22 extension
EOS	end-of-string (0)
opt	additional options for the module (bit field, UNICOM3/4 only): <ul style="list-style-type: none"> bit 0: selects method for halting CORE0: 0: "Halt After PORST", 1: "Halt After RESET", see Remarks bit 1: controls servicing of PORST line: 0: PORST low, UBATT on, PORST high 1: PORST high, UBATT on, PORST low, PORST high other: dummy, should be 0.
key	default setting when not specified is 0x00. Key value for unlocking a locked debug interface. <i>key 1 .. key 32</i> build 8 32-bit-words, MSB first each.
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 MODULE command starts the *DAP* communication on *Aurix* and enables its *OCDS* (On Chip Debug System).
- The Unload Form resets the target device and disables the OCDS this way
- After successful executing of MODULE command, form 2 or 3, *MODULE* interface must be enabled using the *CONFIG_UNICOM(1)* command of *FFCOMBI/UCBASE*.
- If *opt* bit 0 is set to 1, "Halt After RESET" instead of "Halt After PORST" is used for halting CORE0. That ensures a halt directly after execution of the internal start software and before running the user software from flash memory.

4.2 CONFIG_MODULE Command

(UNICOM3/4 only)

With this command, TC_DAP can be configured. In this case, DAP bitrate and file type for FASTFLASH can be adjusted.

Command, Form 1

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5
len	ecu	cmd	slot	DAP br	cks
5	0xC0	4	0..3	1..50	

Command, Form 2

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6
len	ecu	cmd	slot	DAP br	ftype	cks
6	0xC0	4	0..3	1..50	0,4,8	

Command, Form 3, (UNICOM3 Rev.D and UNICOM4 only)

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7
len	ecu	cmd	slot	DAP br	DAP sp	ftype	cks
7	0xC0	4	0..3	1..50	0..7	0,4,8	

len	length of telegram
ecu	target address
cmd	command code
DAP br	bitrate of DAP interface, in units to 100 kBits/s
DAP sp	sampling point delay (Rev.D only). See remarks.
ftype	file type for FASTFLASH, 0: BINARY, 4: SRECORD, 8: INTEL HEX. Default after module start is 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

- The command can be used to speed up the toolbox download (DOWNLOAD(22), ref chapter 6.3 on page 17) and FASTFLASH (ref. chapter 5 on page 13).
- With the *DAP sp* parameter, the sampling point inside a bit which is transferred from target to UNICOM over the DAP interface can be adjusted (delayed) in steps of 12.5 ns. In a noisy environment it can help to make DAP communication more stable. The default value is 3.

Chapter 5

FASTFLASH

The TC_DAP module implements FASTFLASH over DAP ONLY. It supports file types of BINARY, MOTOROLA SRECORD and INTEL HEX. Both FASTFLASH commands X_FASTFLASH(14) and X_FASTFLASH_TAB(15) can be used. Refer to `ucbase.pdf` for more information about FASTFLASH.

FASTFLASH can also be used to verify just programmed flash data. To do so, VERIFY mode must be activated before, and the same FASTFLASH command which has been used for programming must be repeated (ref. chapter 6.10 on page 25).

Chapter 6

TC_DAP Module Commands

The following commands are identical for UNICOM II+ and UNICOM3/4. For UNICOM II+, an *ECU Number* of 0xD0 must be chosen to access the commands of module.

For UNICOM3/4 it is assumed that interface slot 0 is configured for *MODULE* interface, so an *ECU number* of 0x80 must be chosen to access the commands of module in this case.

6.1 TC_DAP::READ_VERSION (2)

This command reports about the version information of the TC_DAP module.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	0xD0/0x80	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	0xD0/0x80					

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

Remarks

- For UNICOM II+, a version string of *TC_DAP module* TC_DAP_{UUUUUUU}V_x.y should be reported.
- For UNICOM3/4, a version string of *TC_DAP module* tc_dap_{UUUUUUU}V_x.y should be reported.

6.2 TC_DAP::GET_LAST_STATE (3)

This command reports the content of the *IOINFO* register and can help to find the reason for a failed other command. It must be executed if another command has failed in order to re-start the debug interface.

Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	0xD0/0x80	3	

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	cmd	IOINFO		cks
5	0xD0/0x80	3	MSB	LSB	

len length of telegram
ecu source address
status result status
IOONFO value if the *IOINFO* register.
cks checksum of telegram

Remarks

- For interpretation of the bits of *IOINFO* please refer to the Reference Manual of TC2xx uC, chapter "OCDS"

6.3 TC_DAP::DOWNLOAD (22)

(UNICOM3/4 only)

With help of this command, an operating system or a toolbox can be downloaded into RAM of Aurix uC.

Command

byte 0	byte 1	byte 2	byte 3	...	byte 4	byte x	
len	ecu	cmd	tbx 1	...	tbx n	eos	
N	0xD0/0x80	22		...		0	

byte x	byte x	
freq	can	

byte x	byte x	byte x	byte x	byte x	byte x	byte x	byte x
pin1		pin2		pin3		pin4	
MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB

byte x	...	byte x	byte x	...	byte N-10	byte N-9	
CAN bitrate			CAN FD bitrate			tdel	
MSB	...	LSB	MSB	...	LSB		

byte N-8	byte N-7	byte N-6	byte N-5	
ed typ 1	ed par1 1		ed par2 1	
	MSB	LSB		

byte N-4	byte N-3	byte N-2	byte N-1	N
ed typ 2	ed par1 2		ed par2 2	cks
	MSB	LSB		

len	length of telegram
ecu	target address
cmd	command code
tbx	filename of toolbox
eos	end-of-string, 0
freq	(optional) crystal frequency in MHz
can	(optional) main CAN index, for CAN toolboxes, otherwise: 0
pin x	(optional) 4 pins to set: high byte: port number, bcd coded, bit 7: level, bit 0..3: pin number inside the port
CAN bitrate	(optional) 32bit value, CAN bitrate in bits/s, for CAN toolboxes only

CAN FD bitrate	(optional) 32bit value, CAN FD bitrate in bits/s, for CAN toolboxes that support CAN_FD
tdel	(optional) Transceiver Delay value, for CAN toolboxes that support CAN_FD
ed type n	(optional) type code for an external device which must be activated or configured by the controller during startup of OS (s. remarks)
ed par1 n	(optional) external device connection and configuration informations (s. remarks)
ed par2 n	(optional) external device connection and configuration informations (s. remarks)
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	0xD0/0x80		

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

Remarks

- Toolbox download works even if a software is programmed into flash with a boot mode header that blocks the hw configuration.
- However, DAP interface can be locked by DEBUG UCB.
- *ed type n* is a unique number (selector) that specifies the type of an external device to be handled. It implies in which way the device is connected to the controller. A table of the supported devices can be found in the documentation of the corresponding OS (*TC2xx_OPS*, *TC3xx_OPS*).
- *ed par1 n* is a two byte parameter which is used to configure the selected device (e.g. connection information). Its concrete meaning depends on the device type (for more information refer the documentation of the corresponding OS).
- *ed par2 n* is a one byte parameter which is used to configure the selected device (e.g. connection information). Its concrete meaning depends on the device type (for more information refer the documentation of the corresponding OS).

6.4 TC_DAP::WRITE_REGISTER (102)

With this command, a 32-bit-register can be written.

Command

byte 0	byte 1	byte 2	byte 3	...	byte 6	
len	ecu	cmd	addr			
11	0xD0/0x80	102	MSB	...	LSB	

	byte 7	...	byte 10	byte 11
	data			cks
	MSB	...	LSB	

len	length of telegram
ecu	target address
cmd	command code
addr	register address
data	data to be written
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	0xD0/0x80		

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

6.5 TC_DAP::READ_REGISTER (103)

With this command, the content of a 32-bit-register can be read.

Command

byte 0	byte 1	byte 2	byte 3	...	byte 6	byte 7
len	ecu	cmd	addr			cks
7	0xD0/0x80	103	MSB	...	LSB	

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

Response

byte 0	byte 1	byte 2	byte 3	...	byte 6	byte 7
len	ecu	status	data			cks
7	0xD0/0x80	0xA0	MSB	...	LSB	

len length of telegram
ecu source address
status result status
data content read from desired register
cks checksum of telegram

6.6 TC_DAP::WRITE_MEM (112)

With this command, RAM locations can be written 32-bit-wise.

Command

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	
len	ecu	cmd	opt	addr			
N=8+n	0xD0/0x80	112	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 here, should be 0
addr	start address in RAM where data are being written. Must be a 32-bit-aligned value
data	data to be written. must be a multiple of 4 bytes
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	0xD0/0x80		

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

Remarks

- Only global RAM addresses can be used for RAM write.

6.7 TC_DAP::READ_MEM (113)

With this command, flash or RAM locations can be read 32-bit-wise

Command

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	
len	ecu	cmd	opt	addr			
10	0xD0/0x80	113	0	MSB	...	LSB	

byte 8	byte 9	byte 10
size		cks
MSB	LSB	

len	length of telegram
ecu	target address
cmd	command code
opt	not used here, should be 0
addr	start address in RAM where data should read. Must be a 32-bit-aligned value
size	number of desired data bytes, must be multiple of 4
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	0xD0/0x80	0xA0		...		

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

Remarks

- For reading flash location, an offset of 0x80000000 (cached read) or 0xA0000000 (uncached read) must be applied.
- Flash can only be read if it is programmed at specified location. If it is blank, and ECC trap occurs, and the debug system blocks. Use the *GET_LAST_STATE(3)* command (ref. chapter 6.2 on page 16 for unblock it again if that happens).
- Only global RAM addresses can be used for RAM read.

6.8 TC_DAP::ERASE_CHECK_FLASH (115)

This command can be used to erase or blank check flash sectors.

Command (form 1, erase/check 1 sector)

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	byte 8
len	ecu	cmd	opt	addr			cks
8	0xD0/0x80	115	0,1	MSB	...	LSB	

Command (form 2, erase/check n sectors)

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	
len	ecu	cmd	opt	addr			
9	0xD0/0x80	115	0,1	MSB	...	LSB	

byte 8	byte 9
sectors	cks

len	length of telegram
ecu	target address
cmd	command code
opt	controls the action: 0: erase, 1: blank check
addr	address of first sector to be erased/checked
sectors	number of sectors to be erased/checked, beginning with this one <i>addr</i> points to. All sectors must be located in the same flash block.
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	0xD0/0x80		

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

6.9 TC_DAP::PROGRAM_FLASH (116)

(UNICOM3/4 only)

With this command, single data pages of flash memory can be programmed.

Command

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	
len	ecu	cmd	opt	addr			
N=8+n	0x80	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 here, should be 0
addr	flash offset (starts with 0x00000000), must be 32-byte-aligned for program flash and 8-byte-aligned for data flash
data	page data, must be a multiple of 32 bytes for program flash and 8 bytes for data flash
cks	checksum of telegram

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	0x80		

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

Remarks

- Flash memory must be in erased state before programming, use ERASE_FLASH(115), ref. chapter 6.8 on page 23.

6.10 TC_DAP::SWITCH_VERIFY (117)

(UNICOM3/4 only)

With this command the behaviour of FASTFLASH can be selected: either programming of data (program mode) or verify of data (verify mode).

Command

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	mode	cks
4	0x80	117	0,1	

len	length of telegram
ecu	target address
cmd	command code
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	0x80		

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

Remarks

- Select the mode before executing an X_FLASHFLASH(14) or X_FASTFLASH_TAB(15) command.

6.11 TC_DAP::READ_FLASH (123)

(UNICOM3/4 only)

With this command, data from flash memory can be read.

Command

byte 0	byte 1	byte 2	byte 3	byte 4	...	byte 7	
len	ecu	cmd	opt	addr			
10	0x80	123	0	MSB	...	LSB	

	byte 8	byte 9	byte 10
	size		cks
	MSB	LSB	

len	length of telegram
ecu	target address
cmd	command code
opt	not used here, should be 0
addr	start address offset in flash (starting with 0x00000000), must be 4-byte-aligned
size	number of bytes to be read, must be a multiple of 4.
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	0x80	0xA0		...		

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

6.12 TC_DAP::GO (126)

(UNICOM3/4 only)

This command lets CORE0 of uC execute code on specified address.

Command

byte 0	byte 1	byte 2	byte 3	...	byte 6	byte 7
len	ecu	cmd	address		cks	
7	0x80	126	MSB	...	LSB	

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

Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	0x80		

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

Remarks

- This command only works if CORE0 is halted, means no OS or toolbox should be downloaded, or even this GO command isn't executed before.

6.13 TC_DAP::TC_TYPE (250)

(UNICOM3/4 only)

This command can be used to read or define the TriCore type of an universal OS (such as tc2xx_os) to be downloaded.

Command (form 1, set TC type)

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	cmd	type	cks
4	0x80	250		

Command (form 2, read configured TC type)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	0x80	250	

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

Response

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	status	type	cks
4	0x80	0xA0		

len length of telegram
ecu source address
status result status
type TC type that is currently configured
cks checksum of telegram

Remarks

- TC type: first two digits of the device type as binary coded decimal (e.g. 0x26 for TC26x devices).
- if this command is not invoked before downloading an universal OS, the OS will start with its default value.

- Selected TC type also influences the DAP ONLY flash programming and erasing algorithms of TC_DAP module, be sure to select the right one!

6.14 TC_DAP::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
DAUGHTERBOARD_ERROR	0xCF	Wrong device type of UNI-COM
PARAMETER_ERROR	0xB0	Wrong parameter in command telegram
LENGTH_ERROR	0xB3	Command telegram from test computer with wrong length
ADDRESS_ERROR	0xB7	Wrong address parameter specified in command telegram
ECU_CHECKSUM_ERROR	0xC2	Wrong DAP checksum received
ECU_TIMEOUT_ERROR	0xC5	No response from target within specified timeout time
ERASE_ERROR	0xD0	Error while erasing flash memory
PROGRAM_ERROR	0xD1	Error while programming flash memory
BLANK_CHECK_ERROR	0xD2	Flash not blank at specified address range
VERIFY_ERROR	0xD3	Error while verifying flash data
FLASH_TIMEOUT_ERROR	0xD7	Flash timed out while operating
NO_VGPIO_ERROR	0xE0	No voltage supply at V_GPIO pin of UNI-COM detected
DAP_SYNC_ERROR	0xE1	Synchronization error while entering DAP mode
INTERFACE_LOCKED_ERROR	0xE2	Debug interface locked. Try MODULE command with key
OCDS_ENABLE_ERROR	0xE3	OCDS couldn't be enabled
CPU_RESET_ERROR	0xE4	CPU software reset trigger failed
CPU_HALT_ERROR	0xE5	CPU could not be halted
EINIT_ERROR	0xE6	End-Of-Init state couldn't be released
UNKNOWN_COMMAND_ERROR	0xFF	Unknown command code received

More error codes can be found in the `ffcombi.pdf` respective in the `ucbase.pdf` documentation file.