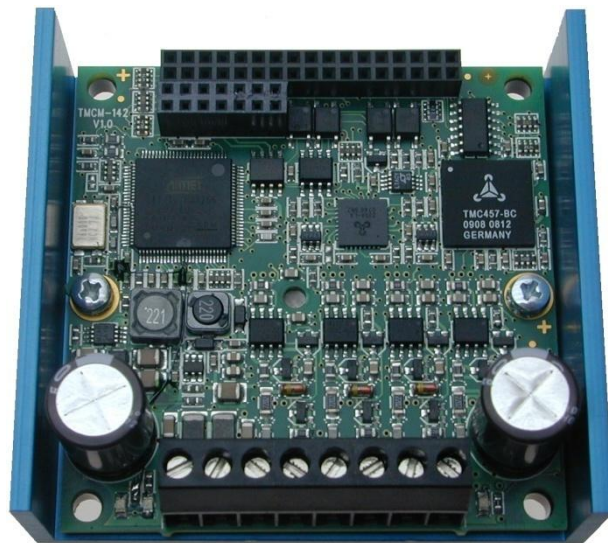


# TMCM-142



## TMCL™ Firmware Manual

Version: 1.01  
March 16<sup>th</sup>, 2009



**TRINAMIC**  
MOTION CONTROL

Trinamic Motion Control GmbH & Co KG  
Sternstraße 67  
D - 20 357 Hamburg, Germany  
<http://www.trinamic.com>

## Table of contents

1	Life support policy .....	4
2	Features .....	5
3	Order codes .....	6
4	Overview .....	7
5	TMCL™ and TMCL-IDE .....	8
5.1	Binary command format .....	8
5.2	Reply format .....	9
5.2.1	Status codes .....	10
5.3	Stand-alone applications .....	10
5.4	TMCL™ command overview .....	10
5.4.1	Motion commands .....	10
5.4.2	Parameter commands .....	11
5.4.3	I/O port commands .....	11
5.4.4	Control commands .....	11
5.4.5	Calculation commands .....	11
5.5	TMCL™ commands .....	12
5.6	The ASCII interface .....	14
5.6.1	Format of the command line .....	14
5.6.2	Format of a reply .....	14
5.6.3	Commands that can be used in ASCII mode .....	14
5.6.4	Configuring the ASCII interface .....	14
5.7	Commands .....	16
5.7.1	ROR (rotate right) .....	16
5.7.2	ROL (rotate left) .....	17
5.7.3	MST (motor stop) .....	18
5.7.4	MVP (move to position) .....	19
5.7.5	SAP (set axis parameter) .....	21
5.7.6	GAP (get axis parameter) .....	24
5.7.7	STAP (store axis parameter) .....	28
5.7.8	RSAP (restore axis parameter) .....	31
5.7.9	SGP (set global parameter) .....	34
5.7.10	GGP (get global parameter) .....	38
5.7.11	STGP (store global parameter) .....	42
5.7.12	RSGP (restore global parameter) .....	44
5.7.13	RFS (reference search) .....	48
5.7.14	SIO (set output) .....	49
5.7.15	GIO (get input/output) .....	50
5.7.16	CALC (calculate) .....	52
5.7.17	COMP (compare) .....	53
5.7.18	JC (jump conditional) .....	54
5.7.19	JA (jump always) .....	55
5.7.20	CSUB (call subroutine) .....	56
5.7.21	RSUB (return from subroutine) .....	57
5.7.22	WAIT (wait for an event to occur) .....	58
5.7.23	STOP (stop TMCL™ program execution) .....	59
5.7.24	SCO (set coordinate) .....	60
5.7.25	GCO (get coordinate) .....	61
5.7.26	CCO (capture coordinate) .....	62
5.7.27	CALCX (calculate using the X register) .....	63
5.7.28	AAP (accumulator to axis parameter) .....	64
5.7.29	AGP (accumulator to global parameter) .....	67
5.7.30	CLE (clear error flags) .....	70
5.7.31	Customer specific TMCL™ command extension (UF0...UF7/user function) .....	71
5.7.32	Request target position reached event .....	72
5.7.33	BIN (return to binary mode) .....	73

5.7.34	TMCL™ Control Functions.....	74
6	Axis parameters .....	76
7	Global parameters .....	79
7.1	Bank 0 .....	79
7.2	Bank 1 .....	81
7.3	Bank 2 .....	82
8	Hints and tips .....	83
8.1	Reference search .....	83
8.2	Fixing microstep errors .....	83
8.3	Using the RS485 interface .....	84
9	Revision history .....	85
9.1	Firmware revision.....	85
9.2	Document Revision .....	85
10	References.....	85

## 1 Life support policy

TRINAMIC Motion Control GmbH & Co. KG does not authorize or warrant any of its products for use in life support systems, without the specific written consent of TRINAMIC Motion Control GmbH & Co. KG.

Life support systems are equipment intended to support or sustain life, and whose failure to perform, when properly used in accordance with instructions provided, can be reasonably expected to result in personal injury or death.

© TRINAMIC Motion Control GmbH & Co. KG 2005

Information given in this data sheet is believed to be accurate and reliable. However neither responsibility is assumed for the consequences of its use nor for any infringement of patents or other rights of third parties, which may result from its use.

Specifications are subject to change without notice.

## 2 Features

The TMC-142 is a high-performance single axis stepper motor controller/driver with encoder feedback. The integrated TMC457 motion controller provides superior performance with regard to microstep resolution (up-to 1024), maximum velocity (integrated chopsync™), ramp calculation (S-shaped ramps, calculated in real-time) and encoder feedback support (closing the loop in hardware with PID regulator). The driver stage supports motors with up-to 5A RMS coil current and offers exceptional low power dissipation.

Together with the TMC-IF standard add-on interface/adaptor board a large number of interface options is available.

### Applications

- Compact high-resolution/high-performance stepper motor controller/driver solutions
- Smooth movements with high microstep resolution and S-shaped ramps
- High precision and high repeatability with encoder feedback and PID position regulator

### Electrical data

- Supply voltage: +18V... +78.5V DC
- Motor current: up-to 7A peak / 5A RMS (programmable)

### Supported motors

- Two phase bipolar motors with 1A to 5A RMS coil current
- Incremental encoder (a/b + optional index channel, differential, open-collector or single ended signals)

### Interfaces

- Optically isolated inputs for home and stop switches
- general purpose analogue and digital inputs and outputs
- RS422, RS232, CAN and USB serial interfaces available
- RS422, RS485, RS232, CAN or USB serial interfaces available on standard add-on interface board TMC-IF

### Features

- 1024 times micro stepping
- Automatic ramp generation (trapezoid and S-shaped) in real-time in hardware
- On the fly alteration of motion parameters (e.g. position, velocity, acceleration)
- Uses TMC457 high performance controller
- Chopsync™ for high speed
- High-efficient operation, low power dissipation
- Integrated protection: overtemperature/undervoltage

### Software

- Stand-alone operation using TMCL™ or remote controlled operation
- Memory for 2048 TMCL™ commands
- PC-based application development software TMCL-IDE included
- CANopen ready

### 3 Order codes

The TMC-142 is currently available with the standard adapter/interface add-on board TCM-IF:

Order code	Description	Dimensions [mm <sup>3</sup> ]
TMC-142-IF	Single axis stepper motor controller/driver, 5A RMS, 75V, with encoder feedback and the standard adapter/interface board TCM-IF	76x70x33

**Table 3.1: Order codes**

Versions without the standard adapter/interface board TCM-IF (just the baseboard) or custom interface boards are available on request.

## 4 Overview

As with most TRINAMIC modules the software running on the microprocessor of the TMC-142 consists of two parts, a boot loader and the firmware itself. Whereas the boot loader is installed during production and testing at TRINAMIC and remains – normally – untouched throughout the whole lifetime, the firmware can be updated by the user. New versions can be downloaded free of charge from the TRINAMIC website (<http://www.trinamic.com>).

The firmware shipped with this module is related to the standard TMCL™ firmware shipped with most of TRINAMIC modules with regard to protocol and commands. Corresponding, this module is based on the TMC457 motion controller for stepper motors and the TMC239 power driver and supports the standard TMCL™ with a special range of values. All commands and parameters available with this unit are explained on the following pages.

## 5 TMCL™ and TMCL-IDE

The TMCM-142 supports TMCL™ direct mode (binary commands or ASCII interface) and stand-alone TMCL™ program execution. You can store up-to 2048 TMCL™ instructions on it.

In direct mode and most cases the TMCL™ communication over RS485, RS232, RS422, USB or CAN follows a strict master/slave relationship. That is, a host computer (e.g. PC/PLC) acting as the interface bus master will send a command to the TMCM-142. The TMCL™ interpreter on the module will then interpret this command, do the initialization of the motion controller, read inputs and write outputs or whatever is necessary according to the specified command. As soon as this step has been done, the module will send a reply back over RS485/RS232/RS422/USB/CAN to the bus master. Only then should the master transfer the next command. Normally, the module will just switch to transmission and occupy the bus for a reply, otherwise it will stay in receive mode. It will not send any data over the interface without receiving a command first. This way, any collision on the bus will be avoided when there are more than two nodes connected to a single bus.

The Trinamic Motion Control Language (TMCL™) provides a set of structured motion control commands. Every motion control command can be given by a host computer or can be stored in an EEPROM on the TMCM™ module to form programs that run stand-alone on the module. For this purpose there are not only motion control commands but also commands to control the program structure (like conditional jumps, compare and calculating).

Every command has a binary representation and a mnemonic. The binary format is used to send commands from the host to a module in direct mode, whereas the mnemonic format is used for easy usage of the commands when developing stand-alone TMCL™ applications using the TMCL-IDE (Integrated Development Environment).

There is also a set of configuration variables for the axis and for global parameters which allow individual configuration of nearly every function of a module. This manual gives a detailed description of all TMCL™ commands and their usage.

### 5.1 Binary command format

Every command has a mnemonic and a binary representation. When commands are sent from a host to a module, the binary format has to be used. Every command consists of a one-byte command field, a one-byte type field, a one-byte motor/bank field and a four-byte value field. So the binary representation of a command always has seven bytes. When a command is to be sent via RS232, RS422, RS485 or USB interface, it has to be enclosed by an address byte at the beginning and a checksum byte at the end. In this case it consists of nine bytes.

This is different when communicating is via the CAN bus. Address and checksum are included in the CAN standard and do not have to be supplied by the user.

**The binary command format for RS232/RS422/RS485/USB is as follows:**

Bytes	Meaning
1	Module address
1	Command number
1	Type number
1	Motor or Bank number
4	Value (MSB first!)
1	Checksum

- The checksum is calculated by adding up all the other bytes using an 8-bit addition.
- When using CAN bus, just leave out the first byte (module address) and the last byte (checksum).

## Checksum calculation

As mentioned above, the checksum is calculated by adding up all bytes (including the module address byte) using 8-bit addition. Here are two examples to show how to do this:

- in C:

```
unsigned char i, Checksum;
unsigned char Command[9];

//Set the "Command" array to the desired command
Checksum = Command[0];
for(i=1; i<8; i++)
    Checksum+=Command[i];

Command[8]=Checksum; //insert checksum as last byte of the command
//Now, send it to the module
```

- in Delphi:

```
var
    i, Checksum: byte;
    Command: array[0..8] of byte;

//Set the "Command" array to the desired command

//Calculate the Checksum:
Checksum:=Command[0];
for i:=1 to 7 do Checksum:=Checksum+Command[i];
Command[8]:=Checksum;
//Now, send the "Command" array (9 bytes) to the module
```

## 5.2 Reply format

Every time a command has been sent to a module, the module sends a reply.

The reply format for RS485/RS422/RS232/USB is as follows:

Bytes	Meaning
1	Reply address
1	Module address
1	Status (e.g. 100 means "no error")
1	Command number
4	Value (MSB first!)
1	Checksum

- The checksum is also calculated by adding up all the other bytes using an 8-bit addition.
- When using CAN bus, the first byte (reply address) and the last byte (checksum) are left out.
- Do not send the next command before you have received the reply!

### 5.2.1 Status codes

The reply contains a status code.

The status code can have one of the following values:

Code	Meaning
100	Successfully executed, no error
101	Command loaded into TMCL™ program EEPROM
1	Wrong checksum
2	Invalid command
3	Wrong type
4	Invalid value
5	Configuration EEPROM locked
6	Command not available

## 5.3 Stand-alone applications

The module is equipped with an EEPROM for storing TMCL™ applications. You can use TMCL-IDE for developing stand-alone TMCL™ applications. You can load them down into the EEPROM and then it will run on the module. The TMCL-IDE contains an editor and a "TMCL™ assembler" where the commands can be entered using their mnemonic format. They will be assembled automatically into their binary representations. Afterwards this code can be downloaded into the module to be executed there.

## 5.4 TMCL™ command overview

In this section a short overview of the TMCL™ commands is given.

### 5.4.1 Motion commands

These commands control the motion of the motor. They are the most important commands and can be used in direct mode or in stand-alone mode.

Mnemonic	Command number	Meaning
ROL	2	Rotate left
ROR	1	Rotate right
MVP	4	Move to position
MST	3	Motor stop
RFS	13	Reference search
SCO	30	Store coordinate
CCO	32	Capture coordinate
GCO	31	Get coordinate

## 5.4.2 Parameter commands

These commands are used to set, read and store axis parameters or global parameters. Axis parameters can be set independently for the axis, whereas global parameters control the behavior of the module itself. These commands can also be used in direct mode and in stand-alone mode.

Mnemonic	Command number	Meaning
SAP	5	Set axis parameter
GAP	6	Get axis parameter
STAP	7	Store axis parameter into EEPROM
RSAP	8	Restore axis parameter from EEPROM
SGP	9	Set global parameter
GGP	10	Get global parameter
STGP	11	Store global parameter into EEPROM
RSGP	12	Restore global parameter from EEPROM

## 5.4.3 I/O port commands

These commands control the external I/O ports and can be used in direct mode and in stand-alone mode.

Mnemonic	Command number	Meaning
SIO	14	Set output
GIO	15	Get input
SAC	29	Access to external SPI device

## 5.4.4 Control commands

These commands are used to control the program flow (loops, conditions, jumps etc.). ***It does not make sense to use them in direct mode. They are intended for stand-alone mode only.***

Mnemonic	Command number	Meaning
JA	22	Jump always
JC	21	Jump conditional
COMP	20	Compare accumulator with constant value
CLE	36	Clear error flags
CSUB	23	Call subroutine
RSUB	24	Return from subroutine
WAIT	27	Wait for a specified event
STOP	28	End of a TMCL™ program

## 5.4.5 Calculation commands

These commands are intended to be used for calculations within TMCL™ applications. ***Although they could also be used in direct mode it does not make much sense to do so.***

Mnemonic	Command number	Meaning
CALC	19	Calculate using the accumulator and a constant value
CALCX	33	Calculate using the accumulator and the X register
AAP	34	Copy accumulator to an axis parameter
AGP	35	Copy accumulator to a global parameter

For calculating purposes there is an accumulator (or accu or A register) and an X register. When executed in a TMCL™ program (in stand-alone mode), all TMCL™ commands that read a value store the result in the accumulator. The X register can be used as an additional memory when doing calculations. It can be loaded from the accumulator.

When a command that reads a value is executed in direct mode the accumulator will not be affected. This means that while a TMCL™ program is running on the module (stand-alone mode), a host can still send commands like GAP, GGP or GIO to the module (e.g. to query the actual position of the motor) without affecting the flow of the TMCL™ program running on the module.

## 5.5 TMCL™ commands

The following TMCL™ commands are currently supported:

Command	Number	Parameter	Description
ROR	1	<velocity>	Rotate right with specified velocity
ROL	2	<velocity>	Rotate left with specified velocity
MST	3		Stop motor movement
MVP	4	ABS REL, <position offset>	Move to position (absolute or relative)
SAP	5	<parameter>, <value>	Set axis parameter (motion control specific settings)
GAP	6	<parameter>	Get axis parameter (read out motion control specific settings)
STAP	7	<parameter>	Store axis parameter permanently (non volatile)
SGP	9	<parameter>, <bank>, <value>	Set global parameter (module specific settings, e.g. communication settings, or TMCL™ user variables)
GGP	10	<parameter>, <bank>	Get global parameter (read out module specific settings e.g. communication settings, or TMCL™ user variables)
STGP	11	<parameter>, <bank>	Store global parameter (TMCL™ user variables only)
RSGP	12	<parameter>, <bank>	Restore global parameter (TMCL™ user variables only)
RFS	13	START STOP STATUS, 0	Reference search
SIO	14	<port>, <bank>, <value>	Set digital output to specified value
GIO	15	<port>, <bank>	Get value of analogue / digital input
CALC	19	<op>, <value>	Process accumulator & value
COMP	20	<value>	Compare accumulator <-> value
JC	21	<condition>, <address>	Conditional jump
JA	22	<address>	Jump absolute
CSUB	23	<address>	Call subroutine
RSUB	24		Return from subroutine

Command	Number	Parameter	Description
WAIT	27	<condition>, <motor>, <ticks>	Wait with further program execution
CALCX	33	<op>	Process accumulator & X-register
AAP	34	<parameter>, <motor>	Accumulator to axis parameter
AGP	35	<parameter>, <bank>	Accumulator to global parameter
STOP	28		Stop program execution

**TMCL™ control commands:**

Command	Number	Parameter	Description
Stop application	128		Stop TMCL™ program execution on module
Run application	129	0 – run from current address <address> – run from specified address	Start TMCL™ program execution
Step application	130		Execute next TMCL™ command only and then stop program execution
Reset application	131		Program execution is stopped and program counter set to zero
Start download mode	132	<start address of application>	Following TMCL™ commands are not interpreted but, transferred to TMCL™ memory
Quit download mode	133		Following TMCL™ commands are interpreted again
Read TMCL memory	134	<memory address>	Read out specified TMCL™ memory location
Get application status	135		Return application status: 0 – stop 1 – run 2 – step 3 – reset
Get firmware version	136	STRING BINARY	Get module / firmware version
Restore factory settings	137	<value=1234>	Restore module to factory defaults

## 5.6 The ASCII interface

Since TMCL™ V3.21 there is also an ASCII interface that can be used to communicate with the module and to send some commands as text strings.

- **The ASCII command line interface is entered by sending the binary command 139 (enter ASCII mode).**
- Afterwards the commands are entered as in the TMCL-IDE. Please note that only those commands, which can be used in direct mode, also can be entered in ASCII mode.
- **For leaving the ASCII mode and re-enter the binary mode enter the command "BIN".**

### 5.6.1 Format of the command line

As the first character, the address character has to be sent. The address character is "A" when the module address is 1, "B" for modules with address 2 and so on. After the address character there may be spaces (but this is not necessary). Then, send the command with its parameters. At the end of a command line a <CR> character has to be sent.

Here are some examples for valid command lines:

```
AMVP ABS, 1, 50000
A MVP ABS, 1, 50000
AROL 2, 500
A MST 1
ABIN
```

These command lines would address the module with address 1. To address e.g. module 3, use address character "C" instead of "A". The last command line shown above will make the module return to binary mode.

### 5.6.2 Format of a reply

After executing the command the module sends back a reply in ASCII format. This reply consists of:

- the address character of the host (host address that can be set in the module)
- the address character of the module
- the status code as a decimal number
- the return value of the command as a decimal number
- a <CR> character

So, after sending AGAP 0, 1 the reply would be BA 100 -5000 if the actual position of axis 1 is -5000, the host address is set to 2 and the module address is 1. The value "100" is the status code 100 that means "command successfully executed".

### 5.6.3 Commands that can be used in ASCII mode

The following commands can be used in ASCII mode: ROL, ROR, MST, MVP, SAP, GAP, STAP, RSAP, SGP, GGP, STGP, RSGP, RFS, SIO, GIO, SAC, SCO, GCO, CCO, UFo, UF1, UF2, UF3, UF4, UF5, UF6, UF7.

There are also special commands that are only available in ASCII mode:

- BIN: This command quits ASCII mode and returns to binary TMCL™ mode.
- RUN: This command can be used to start a TMCL™ program in memory.
- STOP: Stops a running TMCL™ application.

### 5.6.4 Configuring the ASCII interface

The module can be configured so that it starts up either in binary mode or in ASCII mode. **Global parameter 67 is used for this purpose** (please see also chapter 7.1). Bit 0 determines the startup mode: If this bit is set, the module starts up in ASCII mode, else it will start up in binary mode (default). Bit 4 and Bit 5 determine how the characters that are entered are echoed back. Normally, both bits are set to zero. In

this case every character that is entered is echoed back when the module is addressed. A Character can also be erased using the backspace character (press the backspace key in a terminal program). When bit 4 is set and bit 5 is clear the characters that are entered are not echoed back immediately but the entire line will be echoed back after the <CR> character has been sent. When bit 5 is set and bit 4 is clear there will be no echo, only the reply will be sent. This may be useful in RS485 systems.

## 5.7 Commands

The module specific commands are explained in more detail on the following pages. They are listed according to their command number.

### 5.7.1 ROR (rotate right)

With this command the motor will be instructed to rotate with a specified velocity in "right" direction (increasing the position counter).

**Internal function:** First, velocity mode is selected. Then, the velocity value is transferred to axis parameter #0 ("target velocity").

The module is based on the TMC457 motor controller and the TMC239 power driver. This makes possible choosing a velocity between 0 and 2047.

**Related commands:** ROL, MST, SAP, GAP

**Mnemonic:** ROR 0, <velocity>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
1	(don't care)	0*	<velocity> 0... 2147483647

\*motor number is always 0 as only one motor is involved

**Reply in direct mode:**

STATUS	VALUE
100 - OK	(don't care)

**Example:**

Rotate right, motor #0, velocity = 350

*Mnemonic:* ROR 0, 350

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
Value (hex)	\$01	\$01	\$00	\$02	\$00	\$00	\$01	\$5e	\$62

## 5.7.2 ROL (rotate left)

With this command the motor will be instructed to rotate with a specified velocity (opposite direction compared to ROR, decreasing the position counter).

**Internal function:** First, velocity mode is selected. Then, the velocity value is transferred to axis parameter #0 ("target velocity").

The module is based on the TMC457 motor controller and the TMC239 power driver. This makes possible choosing a velocity between 0 and 2047.

**Related commands:** ROR, MST, SAP, GAP

**Mnemonic:** ROL o, <velocity>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
2	(don't care)	o*	<velocity> 0... 2147483647

\*motor number is always 0 as only one motor is involved

**Reply in direct mode:**

STATUS	VALUE
100 - OK	(don't care)

**Example:**

Rotate left, motor #0, velocity = 1200

*Mnemonic:* ROL o, 1200

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
<b>Function</b>	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
<b>Value (hex)</b>	\$01	\$02	\$00	\$00	\$00	\$00	\$04	\$b0	\$b8

### 5.7.3 MST (motor stop)

With this command the motor will be instructed to stop either with deceleration ramp (soft stop) or without (hard stop). Please note: Depending on motor speed a hard stop might lead to step losses.

**Internal function:** The axis parameter "target velocity" is set to zero.

**Related commands:** ROL, ROR, SAP, GAP

**Mnemonic:** MST o

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
3	(don't care)	o*	(don't care)

\*motor number is always 0 as only one motor is involved

**Reply in direct mode:**

STATUS	VALUE
100 – OK	(don't care)

**Example:**

Stop motor  
 Mnemonic: MST o

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
Value (hex)	\$01	\$03	\$00	\$00	\$00	\$00	\$00	\$00	\$05

## 5.7.4 MVP (move to position)

With this command the motor will be instructed to move to a specified relative or absolute position or a pre-programmed coordinate. It will use the acceleration/deceleration ramp and the positioning speed programmed into the unit. This command is non-blocking – that is, a reply will be sent immediately after command interpretation and initialization of the motion controller. Further commands may follow without waiting for the motor reaching its end position. The maximum velocity and acceleration are defined by axis parameters #4 and #5.

### Three operation types are available:

- Moving to an absolute position in the range from -2147483648...+2147483647.
- Starting a relative movement by means of an offset to the actual position. In this case, the new resulting position value must not exceed the above mentioned limits, too.
- Moving the motor to a (previously stored) coordinate (refer to SCO for details).

**Internal function:** A new position value is transferred to the axis parameter #2 target position".

**Related commands:** SAP, GAP, SCO, CCO, GCO, MST

**Mnemonic:** MVP <ABS|REL|COORD>, o, <position|offset|coordinate number>

### Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
4	0 ABS – absolute	o*	<position>
	1 REL – relative	o	<offset>
	2 COORD – coordinate	o	<coordinate number (0..56)>

\*motor number is always 0 as only one motor is involved

### Reply in direct mode:

STATUS	VALUE
100 – OK	(don't care)

### Example:

Move motor to (absolute) position 90000

*Mnemonic:* MVP ABS, o, 9000

### Binary:

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$04	\$00	\$00	\$00	\$01	\$5f	\$90	\$f6

### Example:

Move motor from current position 1000 steps backward (move relative -1000)

*Mnemonic:* MVP REL, o, -1000

### Binary:

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$04	\$01	\$00	\$ff	\$ff	\$fc	\$18	\$18

**Example:**

Move motor to previously stored coordinate #8

*Mnemonic:* MVP COORD, 0, 8

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$04	\$02	\$00	\$00	\$00	\$00	\$08	\$11

- *When moving to a coordinate, the coordinate has to be set properly in advance with the help of the SAP command (parameter 17).*

## 5.7.5 SAP (set axis parameter)

With this command most of the motion control parameters of the module can be specified. The settings will be stored in SRAM and therefore are volatile. That is, information will be lost after power off. **Please use command STAP (store axis parameter) in order to store any setting permanently.**

**Internal function:** The parameter format is converted ignoring leading zeros (or ones for negative values). The parameter is transferred to the correct position in the appropriate device.

**Related commands:** GAP, STAP, RSAP , AAP

**Mnemonic:** SAP <parameter number>, o, <value>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
5	<parameter number>	o*	<value>

\*motor number is always 0 if only one motor is involved

**Reply in direct mode:**

STATUS	VALUE
100 – OK	(don't care)

**Meaning of the letters in column "Access":**

R = readable (GAP)

W = writable (SAP)

E = automatically restored from EEPROM after reset or power-on

**Please note, that for the binary representation <parameter number> has to be filled with the number and the <value> has to be filled with a value from range.**

Number	Axis parameter	Description	Range	Access
0	Target position		-2147483648 .. +2147483647	RW
1	Actual position		-2147483648 .. +2147483647	RW
2	Target speed		-2147483648 .. +2147483647	RW
3	Actual speed		-2147483648 .. +2147483647	RW
4	Max. positioning speed	Speed used for positioning (MVP commands).	0..2147483647	RWE
5	Max. acceleration and deceleration	Sets acceleration and deceleration to the same value.	1..16777215	RWE
6	Max. current	Current when motor is running. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
7	Standby current	Current when motor is standing. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
12	Right limit switch disable	Deactivates the function of the right limit switch when set to 1.	0/1	RWE
13	Left limit switch disable	Deactivates the function of the left limit switch when set to 1.	0/1	RWE
14	Switch mode			RWE
15	Stop deceleration	Deceleration when touching a stop switch.	1..16777215	RWE

Number	Axis parameter	Description	Range	Access
16	Max. acceleration	Acceleration	1..16777215	RWE
17	Max. deceleration	Deceleration	1..16777215	RWE
18	Bow	0: trapezoidal ramps 1..18: S-shaped ramps	0..18	RWE
19	Shaft	Reverses the motor direction when set to 1.	0/1	RWE
20	Standby delay		0..4095	RWE
21	Mixed decay run	0: no mixed decay when running 1: use mixed decay when running	0/1	RWE
22	Mixed decay standby	0: no mixed decay when standing 1: use mixed decay when standing	0/1	RWE
23	Chopper clock divider		96..818	RWE
24	StallGuard threshold	The motor will be stopped when the actual load value exceeds this threshold value. <b>Not supported by TMC-142.</b>	0..7	RWE
27	Microstep resolution	0: 2048 micro steps 1: 1024 micro steps 2: 512 micro steps 3: 256 micro steps 4: 128 micro steps 5: 64 micro steps 6: 32 micro steps 7: 16 micro steps 8: 8 micro steps 9: 4 micro steps 10: 2 micro steps 11: 1 full step	0..11	RWE
28	PID tolerance			RWE
29	Sine wave offset		0..255	RWE
30	PID p factor		0..1677215	RWE
31	PID i factor		0..1677215	RWE
32	PID d factor		0..1677215	RWE
33	PID i clipping		0..32640	RWE
35	PID d clock divider		0..255	RWE
36	PID dv correction		-2147483648 .. +2147483647	RW
37	PID dv clipping		0..2147483647	RWE
40	PID mode	0: do not use PID 1: use PID	0/1	RWE
42	Encoder constant	With every pulse this constant is added to or subtracted from the encoder position register.	0..2147483647	RWE
43	Encoder clear mode			RWE
47	Encoder warn distance	Maximum deviation between motor and encoder.		RWE
48	Compare position	Tbd	-2147483648 .. +2147483647	RWE
50	Right position limit	Position limit when moving in positive direction.	-2147483648 .. +2147483647	RWE
51	Left position limit	Position limit when moving in negative direction.	-2147483648 .. +2147483647	RWE
52	Right position limit enable	The motor cannot drive beyond the right position limit when set to 1.	0/1	RWE
53	Left position limit enable	The motor cannot drive beyond the left position limit when set to 1.	0/1	RWE

Number	Axis parameter	Description	Range	Access
63	Microstep table position		0..8191	RW
64	Step pulse length	Length of the step pulses on the step/direction output.	0..255	RWE
128	Ramp mode	Normally set automatically by the ROL, ROR, MVP and MST commands. 0: positioning mode 1: reserved 2: velocity mode 3: hold mode	0..3	RW
193	Reference search mode			RWE
194	Reference search speed			RWE
195	Reference switch speed			RWE
255	Unit conversion mode	Units to use for velocity and acceleration: 0: use TMC457 units 1: use PPS units	0/1	RWE

**Example:**

set the absolute maximum current of motor to 200mA

*Mnemonic:* SAP 6, 0, 200

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$05	\$06	\$00	\$00	\$00	\$00	\$c8	\$d5

### 5.7.6 GAP (get axis parameter)

Most parameters of the TMCM-142 can be adjusted individually for the axis. With this parameter they can be read out. In stand-alone mode the requested value is also transferred to the accumulator register for further processing purposes (such as conditioned jumps). In direct mode the value read is only output in the "value" field of the reply (without affecting the accumulator).

**Internal function:** The parameter is read out of the correct position in the appropriate device. The parameter format is converted adding leading zeros (or ones for negative values).

**Related commands:** SAP, STAP, AAP, RSAP

**Mnemonic:** GAP <parameter number>, 0

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
6	<parameter number>	0*	(don't care)

\*motor number is always 0 if only one motor is involved

**Reply in direct mode:**

STATUS	VALUE
100 - OK	(don't care)

**Meaning of the letters in column "Access":**

R = readable (GAP)

W = writable (SAP)

E = automatically restored from EEPROM after reset or power-on

**Please note, that for the binary representation <parameter number> has to be filled with the number and the <value> has to be filled with a value from range.**

Number	Axis parameter	Description	Range	Access
0	Target position		-2147483648 .. +2147483647	RW
1	Actual position		-2147483648 .. +2147483647	RW
2	Target speed		-2147483648 .. +2147483647	RW
3	Actual speed		-2147483648 .. +2147483647	RW
4	Max. positioning speed	Speed used for positioning (MVP commands).	0..2147483647	RWE
5	Max. acceleration and deceleration	Sets acceleration and deceleration to the same value.	1..16777215	RWE
6	Max. current	Current when motor is running. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
7	Standby current	Current when motor is standing. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
8	Position reached	1 when the target position and the actual position are equal.	0/1	R
9	Switch status			R
10	Right limit switch status	Logic state of the right switch.	0/1	R
11	Left limit switch status	Logic state of the left switch.	0/1	R
12	Right limit switch disable	Deactivates the function of the right limit switch when set to 1.	0/1	RWE

Number	Axis parameter	Description	Range	Access
13	Left limit switch disable	Deactivates the function of the left limit switch when set to 1.	0/1	RWE
14	Switch mode			RWE
15	Stop deceleration	Deceleration when touching a stop switch.	1..16777215	RWE
16	Max. acceleration	Acceleration	1..16777215	RWE
17	Max. deceleration	Deceleration	1..16777215	RWE
18	Bow	0: trapezoidal ramps 1..18: S-shaped ramps	0..18	RWE
19	Shaft	Reverses the motor direction when set to 1.	0/1	RWE
20	Standby delay		0..4095	RWE
21	Mixed decay run	0: no mixed decay when running 1: use mixed decay when running	0/1	RWE
22	Mixed decay standby	0: no mixed decay when standing 1: use mixed decay when standing	0/1	RWE
23	Chopper clock divider		96..818	RWE
24	StallGuard threshold	The motor will be stopped when the actual load value exceeds this threshold value.	0..7	RWE
25	Actual load value		0..7	R
26	Driver status	Bit 0: driver error Bit 1: over temperature pre-warning Bit 2: motor stall (StallGuard)		R
27	Microstep resolution	0: 2048 micro steps 1: 1024 micro steps 2: 512 micro steps 3: 256 micro steps 4: 128 micro steps 5: 64 micro steps 6: 32 micro steps 7: 16 micro steps 8: 8 micro steps 9: 4 micro steps 10: 2 micro steps 11: 1 full step	0..11	RWE
28	PID tolerance			RWE
29	Sine wave offset		0..255	RWE
30	PID p factor		0..1677215	RWE
31	PID i factor		0..1677215	RWE
32	PID d factor		0..1677215	RWE
33	PID i clipping		0..32640	RWE
34	PID i sum			R
35	PID d clock divider		0..255	RWE
36	PID dv correction		-2147483648 .. +2147483647	RW
37	PID dv clipping		0..2147483647	RWE
38	PID error			R
39	PID Vactual			R
40	PID mode	0: do not use PID 1: use PID	0/1	RWE
41	Encoder position	Actual position of the encoder.	-2147483648 .. +2147483647	R
42	Encoder constant	With every pulse this constant is added to or subtracted from the encoder position register.	0..2147483647	RWE

Number	Axis parameter	Description	Range	Access
43	Encoder clear mode			RWE
44	Encoder status	1 when an encoder null channel event has been detected. Cleared after reading.	0/1	R
45	Encoder latch	Encoder position latched on N channel event.		R
46	Position latch	Motor position latched on stop switch event.		R
47	Encoder warn distance	Maximum deviation between motor and encoder.		RWE
48	Compare position	Tbd	-2147483648 .. +2147483647	RWE
50	Right position limit	Position limit when moving in positive direction.	-2147483648 .. +2147483647	RWE
51	Left position limit	Position limit when moving in negative direction.	-2147483648 .. +2147483647	RWE
52	Right position limit enable	The motor cannot drive beyond the right position limit when set to 1.	0/1	RWE
53	Left position limit enable	The motor cannot drive beyond the left position limit when set to 1.	0/1	RWE
63	Microstep table position		0..8191	RW
64	Step pulse length	Length of the step pulses on the step/direction output.	0..255	RWE
128	Ramp mode	Normally set automatically by the ROL, ROR, MVP and MST commands. 0: positioning mode 1: reserved 2: velocity mode 3: hold mode	0..3	RW
129	Status flags	Bit 0: target position reached (same as parameter 8) Bit 1: target velocity reached (same as parameter 130) Bit 2: motor not moving (v=0) Bit 3: encoder warn distance exceeded		R
130	Velocity reached	Reads 1 when the actual speed is equal to the target speed	0/1	R
193	Reference search mode			RWE
194	Reference search speed			RWE
195	Reference switch speed			RWE
255	Unit conversion mode	Units to use for velocity and acceleration: 0: use TMC457 units 1: use PPS units	0/1	RWE

**Example:**

get the actual position of motor

Mnemonic: GAP 0, 1

Binary:

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$06	\$01	\$00	\$00	\$00	\$00	\$00	\$0a

Reply:

Byte Index	0	1	2	3	4	5	6	7	8
Function	Host-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$02	\$01	\$64	\$06	\$00	\$00	\$02	\$c7	\$36

⇒ **status=no error, position=711**

## 5.7.7 STAP (store axis parameter)

An axis parameter previously set with a *Set Axis Parameter* command (SAP) will be stored permanent. Most parameters are automatically restored after power up.

**Internal function:** An axis parameter value stored in SRAM will be transferred to EEPROM and loaded from EEPROM after next power up.

**Related commands:** SAP, RSAP, GAP, AAP

**Mnemonic:** STAP <parameter number>, 0

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
7	<parameter number>	0* <sup>1</sup>	(don't care)* <sup>2</sup>

\*<sup>1</sup>motor number is always 0 if only one motor is involved

\*<sup>2</sup>the value operand of this function has no effect. Instead, the currently used value (e.g. selected by SAP) is saved.

**Reply in direct mode:**

STATUS	VALUE
100 – OK	(don't care)

**Parameter ranges:**

Parameter number	Motor number	Value
s. chapter 6	0	s. chapter 6

**Meaning of the letters in column "Access":**

R = readable (GAP)

W = writable (SAP)

E = automatically restored from EEPROM after reset or power-on

**Please note, that for the binary representation <parameter number> has to be filled with the number and the <value> has to be filled with a value from range.**

Number	Axis parameter	Description	Range	Access
0	Target position		-2147483648 .. +2147483647	RW
1	Actual position		-2147483648 .. +2147483647	RW
2	Target speed		-2147483648 .. +2147483647	RW
3	Actual speed		-2147483648 .. +2147483647	RW
4	Max. positioning speed	Speed used for positioning (MVP commands).	0..2147483647	RWE
5	Max. acceleration and deceleration	Sets acceleration and deceleration to the same value.	1..16777215	RWE
6	Max. current	Current when motor is running. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
7	Standby current	Current when motor is standing. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
12	Right limit switch disable	Deactivates the function of the right limit switch when set to 1.	0/1	RWE
13	Left limit switch disable	Deactivates the function of the left limit switch when set to 1.	0/1	RWE

Number	Axis parameter	Description	Range	Access
14	Switch mode			RWE
15	Stop deceleration	Deceleration when touching a stop switch.	1..16777215	RWE
16	Max. acceleration	Acceleration	1..16777215	RWE
17	Max. deceleration	Deceleration	1..16777215	RWE
18	Bow	0: trapezoidal ramps 1..18: S-shaped ramps	0..18	RWE
19	Shaft	Reverses the motor direction when set to 1.	0/1	RWE
20	Standby delay		0..4095	RWE
21	Mixed decay run	0: no mixed decay when running 1: use mixed decay when running	0/1	RWE
22	Mixed decay standby	0: no mixed decay when standing 1: use mixed decay when standing	0/1	RWE
23	Chopper clock divider		96..818	RWE
24	StallGuard threshold	The motor will be stopped when the actual load value exceeds this threshold value.	0..7	RWE
27	Microstep resolution	0: 2048 micro steps 1: 1024 micro steps 2: 512 micro steps 3: 256 micro steps 4: 128 micro steps 5: 64 micro steps 6: 32 micro steps 7: 16 micro steps 8: 8 micro steps 9: 4 micro steps 10: 2 micro steps 11: 1 full step	0..11	RWE
28	PID tolerance			RWE
29	Sine wave offset		0..255	RWE
30	PID p factor		0..1677215	RWE
31	PID i factor		0..1677215	RWE
32	PID d factor		0..1677215	RWE
33	PID i clipping		0..32640	RWE
35	PID d clock divider		0..255	RWE
36	PID dv correction		-2147483648 .. +2147483647	RW
37	PID dv clipping		0..2147483647	RWE
40	PID mode	0: do not use PID 1: use PID	0/1	RWE
42	Encoder constant	With every pulse this constant is added to or subtracted from the encoder position register.	0..2147483647	RWE
43	Encoder clear mode			RWE
47	Encoder warn distance	Maximum deviation between motor and encoder.		RWE
48	Compare position	Tbd	-2147483648 .. +2147483647	RWE
50	Right position limit	Position limit when moving in positive direction.	-2147483648 .. +2147483647	RWE
51	Left position limit	Position limit when moving in negative direction.	-2147483648 .. +2147483647	RWE
52	Right position limit enable	The motor cannot drive beyond the right position limit when set to 1.	0/1	RWE

Number	Axis parameter	Description	Range	Access
53	Left position limit enable	The motor cannot drive beyond the left position limit when set to 1.	0/1	RWE
63	Microstep table position		0..8191	RW
64	Step pulse length	Length of the step pulses on the step/direction output.	0..255	RWE
128	Ramp mode	Normally set automatically by the ROL, ROR, MVP and MST commands. 0: positioning mode 1: reserved 2: velocity mode 3: hold mode	0..3	RW
193	Reference search mode			RWE
194	Reference search speed			RWE
195	Reference switch speed			RWE
255	Unit conversion mode	Units to use for velocity and acceleration: 0: use TMC457 units 1: use PPS units	0/1	RWE

**Example:**

store the maximum speed of motor #0  
Mnemonic: STAP 4, 0

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$07	\$04	\$00	\$00	\$00	\$00	\$00	\$0d

**Note:** The STAP command will not have any effect when the configuration EEPROM is locked (refer to 7.1). In direct mode, the error code 5 (configuration EEPROM locked, see also section 5.2.1) will be returned in this case.

## 5.7.8 RSAP (restore axis parameter)

For all configuration-related axis parameters non-volatile memory locations are provided. By default, most parameters are automatically restored after power up. A single parameter that has been changed before can be reset by this instruction also.

**Internal function:** The specified parameter is copied from the configuration EEPROM memory to its RAM location.

**Relate commands:** SAP, STAP, GAP, AAP

**Mnemonic:** RSAP <parameter number>, 0

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
8	<parameter number>	0*	(don't care)

\*motor number is always 0 if only one motor is involved

**Reply structure in direct mode:**

STATUS	VALUE
100 – OK	(don't care)

**List of basic parameters, which can be used for RSAP:**

**Meaning of the letters in column "Access":**

R = readable (GAP)

W = writable (SAP)

**Please note, that for the binary representation <parameter number> has to be filled with the number and the binary representation <value> has to be filled with a value from range.**

Number	Axis parameter	Description	Range	Access
0	Target position		-2147483648 .. +2147483647	RW
1	Actual position		-2147483648 .. +2147483647	RW
2	Target speed		-2147483648 .. +2147483647	RW
3	Actual speed		-2147483648 .. +2147483647	RW
4	Max. positioning speed	Speed used for positioning (MVP commands).	0..2147483647	RWE
5	Max. acceleration and deceleration	Sets acceleration and deceleration to the same value.	1..16777215	RWE
6	Max. current	Current when motor is running. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
7	Standby current	Current when motor is standing. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
12	Right limit switch disable	Deactivates the function of the right limit switch when set to 1.	0/1	RWE
13	Left limit switch disable	Deactivates the function of the left limit switch when set to 1.	0/1	RWE
14	Switch mode			RWE
15	Stop deceleration	Deceleration when touching a stop switch.	1..16777215	RWE

Number	Axis parameter	Description	Range	Access
16	Max. acceleration	Acceleration	1..16777215	RWE
17	Max. deceleration	Deceleration	1..16777215	RWE
18	Bow	0: trapezoidal ramps 1..18: S-shaped ramps	0..18	RWE
19	Shaft	Reverses the motor direction when set to 1.	0/1	RWE
20	Standby delay		0..4095	RWE
21	Mixed decay run	0: no mixed decay when running 1: use mixed decay when running	0/1	RWE
22	Mixed decay standby	0: no mixed decay when standing 1: use mixed decay when standing	0/1	RWE
23	Chopper clock divider		96..818	RWE
24	StallGuard threshold	The motor will be stopped when the actual load value exceeds this threshold value.	0..7	RWE
27	Microstep resolution	0: 2048 micro steps 1: 1024 micro steps 2: 512 micro steps 3: 256 micro steps 4: 128 micro steps 5: 64 micro steps 6: 32 micro steps 7: 16 micro steps 8: 8 micro steps 9: 4 micro steps 10: 2 micro steps 11: 1 full step	0..11	RWE
28	PID tolerance			RWE
29	Sine wave offset		0..255	RWE
30	PID p factor		0..1677215	RWE
31	PID i factor		0..1677215	RWE
32	PID d factor		0..1677215	RWE
33	PID i clipping		0..32640	RWE
35	PID d clock divider		0..255	RWE
36	PID dv correction		-2147483648 .. +2147483647	RW
37	PID dv clipping		0..2147483647	RWE
40	PID mode	0: do not use PID 1: use PID	0/1	RWE
42	Encoder constant	With every pulse this constant is added to or subtracted from the encoder position register.	0..2147483647	RWE
43	Encoder clear mode			RWE
47	Encoder warn distance	Maximum deviation between motor and encoder.		RWE
48	Compare position	Tbd	-2147483648 .. +2147483647	RWE
50	Right position limit	Position limit when moving in positive direction.	-2147483648 .. +2147483647	RWE
51	Left position limit	Position limit when moving in negative direction.	-2147483648 .. +2147483647	RWE
52	Right position limit enable	The motor cannot drive beyond the right position limit when set to 1.	0/1	RWE
53	Left position limit enable	The motor cannot drive beyond the left position limit when set to 1.	0/1	RWE
63	Microstep table position		0..8191	RW

Number	Axis parameter	Description	Range	Access
64	Step pulse length	Length of the step pulses on the step/direction output.	0..255	RWE
128	Ramp mode	Normally set automatically by the ROL, ROR, MVP and MST commands. 0: positioning mode 1: reserved 2: velocity mode 3: hold mode	0..3	RW
193	Reference search mode			RWE
194	Reference search speed			RWE
195	Reference switch speed			RWE
255	Unit conversion mode	Units to use for velocity and acceleration: 0: use TMC457 units 1: use PPS units	0/1	RWE

**Example:**

restore the maximum current of motor #0  
Mnemonic: RSAP 6, 0

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$08	\$06	\$00	\$00	\$00	\$00	\$00	\$10

## 5.7.9 SGP (set global parameter)

With this command most of the module specific parameters not directly related to motion control can be specified and the TMCL™ user variables can be changed. Global parameters are related to the host interface, peripherals or application specific variables. The different groups of these parameters are organized in "banks" to allow a larger total number for future products. Currently, only bank 0 and 1 are used for global parameters, and bank 2 is used for user variables.

**All module settings will automatically be stored non-volatile (internal EEPROM of the processor). The TMCL™ user variables will not be stored in the EEPROM automatically, but this can be done by using STGP commands.**

**Internal function:** the parameter format is converted ignoring leading zeros (or ones for negative values). The parameter is transferred to the correct position in the appropriate (on board) device.

**Related commands:** GGP, STGP, RSGP, AGP

**Mnemonic:** SGP <parameter number>, <bank number>, <value>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
9	<parameter number>	<bank number>	<value>

**Reply in direct mode:**

STATUS	VALUE
100 - OK	(don't care)

**Global parameters of bank 0, which can be used for SGP:**

**Meaning of the letters in column "Access":**

- R = readable (GGP)
- W = writeable (SGP)
- E = automatically restored from EEPROM after reset or power-on.

Number	Global parameter	Description	Range	Access
64	EEPROM magic	Setting this parameter to a different value as \$E4 will cause re-initialization of the axis and global parameters (to factory defaults) after the next power up. This is useful in case of miss-configuration.	0...255	RWE
65	RS232/RS485 baud rate	0 - 9600 baud (default) 1 - 14400 baud 2 - 19200 baud 3 - 28800 baud 4 - 38400 baud 5 - 57600 baud 6 - 76800 baud <b>Not supported by Windows!</b> 7 - 115200 baud <b>115200 does not work with most host PCs as the baud rate error on the module is too high (-3.5% baud rate error).</b>	0...7	RWE
66	Serial address	The module (target) address for RS-232/RS-485.	0...255	RWE

Number	Global parameter	Description	Range	Access
67	ASCII mode	Configure the TMCL™ ASCII interface: Bit 0: 0 – start up in binary (normal) mode 1 – start up in ASCII mode Bits 4 and 5: 00 – Echo back each character 01 – Echo back complete command 10 – Do not send echo, only send command reply		RWE
69	CAN bit rate	1 – 10kBit/s 2 – 20kBit/s 3 – 50kBit/s 4 – 100kBit/s 5 – 125kBit/s 6 – 250kBit/s (default) 7 – 500kBit/s 8 – 1000kBit/s	1..7	RWE
70	CAN reply ID	The CAN ID for replies from the board (default: 2)	0..7ff	RWE
71	CAN ID	The module (target) address for CAN (default: 1)	0..7ff	RWE
73	Configuration EEPROM lock flag	Write: 1234 to lock the EEPROM, 4321 to unlock it. Read: 1=EEPROM locked, 0=EEPROM unlocked.	0/1	RWE
75	Telegram pause time	Pause time before the reply via RS232 or RS485 is sent. For RS232 set to 0. For RS485 it is often necessary to set it to 15 (for RS485 adapters controlled by the RTS pin). For CAN interface this parameter has no effect!	0...255	RWE
76	Serial host address	Host address used in the reply telegrams sent back via RS232 or RS485.	0..255	RWE
77	Auto start mode	0: Do not start TMCL™ application after power up (default). 1: Start TMCL™ application automatically after power up.	0/1	RWE
80	Shutdown pin functionality	Select the functionality of the SHUTDOWN pin 0 – no function 1 – high active 2 – low active	0..2	RWE
81	TMCL™ code protection	Protect a TMCL™ program against disassembling or overwriting. 0 – no protection 1 – protection against disassembling 2 – protection against overwriting 3 – protection against disassembling and overwriting <b><i>If you switch off the protection against disassembling, the program will be erased first! Changing this value from 1 or 3 to 0 or 2, the TMCL™ program will be wiped off.</i></b>	0,1,2,3	RWE
132	Tick timer	A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value.		RW

#### **Global parameters of bank 1, which can be used for SGP:**

The global parameter bank 1 can be used for customer specific extensions of the firmware. Such extensions will be done by Trinamic on customer request.

#### **Global parameters of bank 2, which can be used for SGP:**

Bank 2 contains general purpose 32 bit variables for the use in TMCL™ applications. They are located in RAM and can be stored to EEPROM. After booting, their values are automatically restored to the RAM.

Number	Global parameter	Description	Range	Access
0	General purpose variable #0	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
1	General purpose variable #1	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
2	General purpose variable #2	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
3	General purpose variable #3	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
4	General purpose variable #4	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
5	General purpose variable #5	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
6	General purpose variable #6	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
7	General purpose variable #7	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
8	General purpose variable #8	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
9	General purpose variable #9	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
10	General purpose variable #10	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
11	General purpose variable #11	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
12	General purpose variable #12	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
13	General purpose variable #13	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
14	General purpose variable #14	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
15	General purpose variable #15	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
16	General purpose variable #16	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
17	General purpose variable #17	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
18	General purpose variable #18	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
19	General purpose variable #19	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
20..55	General purpose variables #20..#55	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE

*Please refer to chapter 7 for more information about bank 0 to 2.*

**Example:**

set the serial address of the target device to 3  
*Mnemonic: SGP 66, 0, 3*

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$09	\$42	\$00	\$00	\$00	\$00	\$03	\$4f

### 5.7.10 GGP (get global parameter)

All global parameters can be read with this function. Global parameters are related to the host interface, peripherals or application specific variables. The different groups of these parameters are organized in "banks" to allow a larger total number for future products. Currently, only bank 0 and 1 are used for global parameters, and bank 2 is used for user variables. Please refer to chapter 7 for a complete parameter list.

**Internal function:** The parameter is read out of the correct position in the appropriate device. The parameter format is converted adding leading zeros (or ones for negative values).

**Related commands:** SGP, STGP, RSGP, AGP

**Mnemonic:** GGP <parameter number>, <bank number>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
10	(see chapter 7)	<bank number> see chapter 7	(don't care)

**Reply in direct mode:**

STATUS	VALUE
100 – OK	(don't care)

**Global parameters of bank 0, which can be used for GGP:**

**Meaning of the letters in column "Access":**

- R = readable (GGP)
- W = writeable (SGP)
- E = automatically restored from EEPROM after reset or power-on.

Number	Global parameter	Description	Range	Access
64	EEPROM magic	Setting this parameter to a different value as \$E4 will cause re-initialization of the axis and global parameters (to factory defaults) after the next power up. This is useful in case of miss-configuration.	0...255	RWE
65	RS232/RS485 baud rate	0 – 9600 baud (default) 1 – 14400 baud 2 – 19200 baud 3 – 28800 baud 4 – 38400 baud 5 – 57600 baud 6 – 76800 baud <b>Not supported by Windows!</b> 7 – 115200 baud <b>115200 does not work with most host PCs as the baud rate error on the module is too high (-3.5% baud rate error).</b>	0...7	RWE
66	Serial address	The module (target) address for RS-232/RS-485.	0...255	RWE
67	ASCII mode	Configure the TMCL™ ASCII interface: Bit 0: 0 – start up in binary (normal) mode 1 – start up in ASCII mode Bits 4 and 5: 00 – Echo back each character 01 – Echo back complete command 10 – Do not send echo, only send command reply		RWE

Number	Global parameter	Description	Range	Access
69	CAN bit rate	1 – 10kBit/s 2 – 20kBit/s 3 – 50kBit/s 4 – 100kBit/s 5 – 125kBit/s 6 – 250kBit/s (default) 7 – 500kBit/s 8 – 1000kBit/s	1..7	RWE
70	CAN reply ID	The CAN ID for replies from the board (default: 2)	0..7ff	RWE
71	CAN ID	The module (target) address for CAN (default: 1)	0..7ff	RWE
73	Configuration EEPROM lock flag	Write: 1234 to lock the EEPROM, 4321 to unlock it. Read: 1=EEPROM locked, 0=EEPROM unlocked.	0/1	RWE
75	Telegram pause time	Pause time before the reply via RS232 or RS485 is sent. For RS232 set to 0. For RS485 it is often necessary to set it to 15 (for RS485 adapters controlled by the RTS pin). For CAN interface this parameter has no effect!	0...255	RWE
76	Serial host address	Host address used in the reply telegrams sent back via RS232 or RS485.	0..255	RWE
77	Auto start mode	0: Do not start TMCL™ application after power up (default). 1: Start TMCL™ application automatically after power up.	0/1	RWE
80	Shutdown pin functionality	Select the functionality of the SHUTDOWN pin 0 – no function 1 – high active 2 – low active	0..2	RWE
81	TMCL™ code protection	Protect a TMCL™ program against disassembling or overwriting. 0 – no protection 1 – protection against disassembling 2 – protection against overwriting 3 – protection against disassembling and overwriting <b><i>If you switch off the protection against disassembling, the program will be erased first! Changing this value from 1 or 3 to 0 or 2, the TMCL™ program will be wiped off.</i></b>	0,1,2,3	RWE
128	TMCL™ application status	0 – stop 1 – run 2 – step 3 – reset	0..3	R
129	Download mode	0 – normal mode 1 – download mode	0/1	R
130	TMCL™ program counter	The index of the currently executed TMCL™ instruction.		R
132	Tick timer	A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value.		RW

#### **Global parameters of bank 1, which can be used for GGP:**

The global parameter bank 1 contains can be used for customer specific extensions of the firmware. Such extensions will be done by Trinamic on customer request.

#### **Global parameters of bank 2, which can be used for GGP:**

Bank 2 contains general purpose 32 bit variables for the use in TMCL™ applications. They are located in RAM and can be stored to EEPROM. After booting, their values are automatically restored to the RAM.

Number	Global parameter	Description	Range	Access
0	General purpose variable #0	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
1	General purpose variable #1	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
2	General purpose variable #2	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
3	General purpose variable #3	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
4	General purpose variable #4	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
5	General purpose variable #5	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
6	General purpose variable #6	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
7	General purpose variable #7	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
8	General purpose variable #8	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
9	General purpose variable #9	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
10	General purpose variable #10	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
11	General purpose variable #11	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
12	General purpose variable #12	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
13	General purpose variable #13	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
14	General purpose variable #14	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
15	General purpose variable #15	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
16	General purpose variable #16	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
17	General purpose variable #17	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
18	General purpose variable #18	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
19	General purpose variable #19	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
20..55	General purpose variables #20..#55	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE

*Please refer to chapter 7 for more information about bank 0 to 2.*

**Example:**

get the serial address of the target device

*Mnemonic: GGP 66, 0**Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$0a	\$42	\$00	\$00	\$00	\$00	\$00	\$4d

*Reply:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Host-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$02	\$01	\$64	\$0a	\$00	\$00	\$00	\$01	\$72

⇒ **Status=no error, Value=1**

### 5.7.11 STGP (store global parameter)

This command is used to store TMCL™ user variables permanently in the EEPROM of the module. Some global parameters are located in RAM memory, so without storing modifications are lost at power down. This instruction enables enduring storing. Most parameters are automatically restored after power up (see the list of global parameters in chapter 7).

**Internal function:** The specified parameter is copied from its RAM location to the configuration EEPROM.

**Related commands:** SGP, GGP, RSGP, AGP

**Mnemonic:** STGP <parameter number>, <bank number>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
11	(see chapter 8)	<bank number> (see chapter 8)	(don't care)

**Reply in direct mode:**

STATUS	VALUE
100 – OK	(don't care)

**Global parameters of bank 0, which can be used for STGP:**

**Meaning of the letters in column "Access":**

- R = readable (GGP)
- W = writeable (SGP)
- E = automatically restored from EEPROM after reset or power-on.

Number	Global parameter	Description	Range	Access
64	EEPROM magic	Setting this parameter to a different value as \$E4 will cause re-initialization of the axis and global parameters (to factory defaults) after the next power up. This is useful in case of miss-configuration.	0...255	RWE
65	RS232/RS485 baud rate	0 – 9600 baud (default) 1 – 14400 baud 2 – 19200 baud 3 – 28800 baud 4 – 38400 baud 5 – 57600 baud 6 – 76800 baud <b>Not supported by Windows!</b> 7 – 115200 baud <b>115200 does not work with most host PCs as the baud rate error on the module is too high (-3.5% baud rate error).</b>	0...7	RWE
66	Serial address	The module (target) address for RS-232/RS-485.	0...255	RWE
67	ASCII mode	Configure the TMCL™ ASCII interface: Bit 0: 0 – start up in binary (normal) mode 1 – start up in ASCII mode Bits 4 and 5: 00 – Echo back each character 01 – Echo back complete command 10 – Do not send echo, only send command reply		RWE

Number	Global parameter	Description	Range	Access
69	CAN bit rate	1 – 10kBit/s 2 – 20kBit/s 3 – 50kBit/s 4 – 100kBit/s 5 – 125kBit/s 6 – 250kBit/s (default) 7 – 500kBit/s 8 – 1000kBit/s	1..7	RWE
70	CAN reply ID	The CAN ID for replies from the board (default: 2)	0..7ff	RWE
71	CAN ID	The module (target) address for CAN (default: 1)	0..7ff	RWE
73	Configuration EEPROM lock flag	Write: 1234 to lock the EEPROM, 4321 to unlock it. Read: 1=EEPROM locked, 0=EEPROM unlocked.	0/1	RWE
75	Telegram pause time	Pause time before the reply via RS232 or RS485 is sent. For RS232 set to 0. For RS485 it is often necessary to set it to 15 (for RS485 adapters controlled by the RTS pin). For CAN interface this parameter has no effect!	0...255	RWE
76	Serial host address	Host address used in the reply telegrams sent back via RS232 or RS485.	0..255	RWE
77	Auto start mode	0: Do not start TMCL™ application after power up (default). 1: Start TMCL™ application automatically after power up.	0/1	RWE
80	Shutdown pin functionality	Select the functionality of the SHUTDOWN pin 0 – no function 1 – high active 2 – low active	0..2	RWE
81	TMCL™ code protection	Protect a TMCL™ program against disassembling or overwriting. 0 – no protection 1 – protection against disassembling 2 – protection against overwriting 3 – protection against disassembling and overwriting <b><i>If you switch off the protection against disassembling, the program will be erased first! Changing this value from 1 or 3 to 0 or 2, the TMCL™ program will be wiped off.</i></b>	0,1,2,3	RWE
132	Tick timer	A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value.		RW

#### **Global parameters of bank 1, which can be used for STGP:**

The global parameter bank 1 can be used in customer specific extensions of the firmware. Such extensions will be done by Trinamic on customer request.

#### **Global parameters of bank 2, which can be used for STGP:**

Bank 2 contains general purpose 32 bit variables for the use in TMCL™ applications. They are located in RAM and can be stored to EEPROM. After booting, their values are automatically restored to the RAM.

Number	Global parameter	Description	Range	Access
0	General purpose variable #0	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
1	General purpose variable #1	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
2	General purpose variable #2	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
3	General purpose variable #3	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
4	General purpose variable #4	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE

Number	Global parameter	Description	Range	Access
5	General purpose variable #5	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
6	General purpose variable #6	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
7	General purpose variable #7	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
8	General purpose variable #8	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
9	General purpose variable #9	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
10	General purpose variable #10	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
11	General purpose variable #11	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
12	General purpose variable #12	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
13	General purpose variable #13	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
14	General purpose variable #14	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
15	General purpose variable #15	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
16	General purpose variable #16	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
17	General purpose variable #17	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
18	General purpose variable #18	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
19	General purpose variable #19	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
20..55	General purpose variables #20..#55	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE

**Example:**

store the serial address of the target device  
*Mnemonic:* STGP 42, 0

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$0b	\$42	\$00	\$00	\$00	\$00	\$00	\$4e

**Note:** The STAP command will not have any effect when the configuration EEPROM is locked (refer to 7.1). In direct mode, the error code 5 (configuration EEPROM locked, see also section 5.2.1) will be returned in this case. Please refer to chapter 7 for more information about bank 0 to 2.

### 5.7.12 RSGP (restore global parameter)

With this command the contents of a TMCL™ user variable can be restored from the EEPROM. For all configuration-related axis parameters, non-volatile memory locations are provided. By default, most parameters are automatically restored after power up (see axis parameter list in chapter 7). A single parameter that has been changed before can be reset by this instruction.

**Internal function:** The specified parameter is copied from the configuration EEPROM memory to its RAM location.

**Relate commands:** SAP, STAP, GAP, AAP

**Mnemonic:** RSAP <parameter number>, 0

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
8	<parameter number>	0*	(don't care)

\*motor number is always 0 if only one motor is involved

**Reply structure in direct mode:**

STATUS	VALUE
100 – OK	(don't care)

**Global parameters of bank 0, which can be used for RSGP:****Meaning of the letters in column "Access":**

- R = readable (GGP)
- W = writeable (SGP)
- E = automatically restored from EEPROM after reset or power-on.

Number	Global parameter	Description	Range	Access
64	EEPROM magic	Setting this parameter to a different value as \$E4 will cause re-initialization of the axis and global parameters (to factory defaults) after the next power up. This is useful in case of miss-configuration.	0...255	RWE
65	RS232/RS485 baud rate	0 – 9600 baud (default) 1 – 14400 baud 2 – 19200 baud 3 – 28800 baud 4 – 38400 baud 5 – 57600 baud 6 – 76800 baud <b>Not supported by Windows!</b> 7 – 115200 baud <b>115200 does not work with most host PCs as the baud rate error on the module is too high (-3.5% baud rate error).</b>	0...7	RWE
66	Serial address	The module (target) address for RS-232/RS-485.	0...255	RWE
67	ASCII mode	Configure the TMCL™ ASCII interface: Bit 0: 0 – start up in binary (normal) mode 1 – start up in ASCII mode Bits 4 and 5: 00 – Echo back each character 01 – Echo back complete command 10 – Do not send echo, only send command reply		RWE
69	CAN bit rate	1 – 10kBit/s 2 – 20kBit/s 3 – 50kBit/s 4 – 100kBit/s 5 – 125kBit/s 6 – 250kBit/s (default) 7 – 500kBit/s 8 – 1000kBit/s	1..7	RWE
70	CAN reply ID	The CAN ID for replies from the board (default: 2)	0..7ff	RWE
71	CAN ID	The module (target) address for CAN (default: 1)	0..7ff	RWE
73	Configuration EEPROM lock flag	Write: 1234 to lock the EEPROM, 4321 to unlock it. Read: 1=EEPROM locked, 0=EEPROM unlocked.	0/1	RWE
75	Telegram pause time	Pause time before the reply via RS232 or RS485 is sent. For RS232 set to 0. For RS485 it is often necessary to set it to 15 (for RS485 adapters controlled by the RTS pin). For CAN interface this parameter has no effect!	0...255	RWE
76	Serial host address	Host address used in the reply telegrams sent back via RS232 or RS485.	0..255	RWE
77	Auto start mode	0: Do not start TMCL™ application after power up (default). 1: Start TMCL™ application automatically after power up.	0/1	RWE
80	Shutdown pin functionality	Select the functionality of the SHUTDOWN pin 0 – no function 1 – high active 2 – low active	0..2	RWE

Number	Global parameter	Description	Range	Access
81	TMCL™ code protection	Protect a TMCL™ program against disassembling or overwriting. 0 – no protection 1 – protection against disassembling 2 – protection against overwriting 3 – protection against disassembling and overwriting <b><i>If you switch off the protection against disassembling, the program will be erased first! Changing this value from 1 or 3 to 0 or 2, the TMCL™ program will be wiped off.</i></b>	0,1,2,3	RWE
132	Tick timer	A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value.		RW

**Global parameters of bank 1, which can be used for RSGP:**

The global parameter bank 1 can be used in customer specific extensions of the firmware. Such extensions are done by Trinamic on customer request.

**Global parameters of bank 2, which can be used for RSGP:**

Bank 2 contains general purpose 32 bit variables for the use in TMCL™ applications. They are located in RAM and can be stored to EEPROM. After booting, their values are automatically restored to the RAM.

Number	Global parameter	Description	Range	Access
0	General purpose variable #0	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
1	General purpose variable #1	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
2	General purpose variable #2	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
3	General purpose variable #3	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
4	General purpose variable #4	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
5	General purpose variable #5	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
6	General purpose variable #6	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
7	General purpose variable #7	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
8	General purpose variable #8	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
9	General purpose variable #9	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
10	General purpose variable #10	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
11	General purpose variable #11	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
12	General purpose variable #12	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
13	General purpose variable #13	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
14	General purpose variable #14	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
15	General purpose variable #15	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
16	General purpose variable #16	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
17	General purpose variable #17	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
18	General purpose variable #18	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
19	General purpose variable #19	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
20..55	General purpose variables #20..#55	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE

**Please refer to chapter 7 for more information about bank 0 to 2.**

**Example:**

restore the maximum current of motor #0  
Mnemonic: RSAP 6, 0

Binary:

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$08	\$06	\$00	\$00	\$00	\$00	\$00	\$10



### 5.7.13 RFS (reference search)

The PD-140-42-SE has a built-in reference search algorithm which can be used. The reference search algorithm provides switching point calibration and three switch modes. The status of the reference search can also be queried to see if it has already finished. (In a TMCL program it is better to use the WAIT command to wait for the end of a reference search.) Please see the appropriate parameters in the axis parameter table to configure the reference search algorithm to meet your needs (chapter6). The reference search can be started, stopped, and the actual status of the reference search can be checked.

**Internal function:** The reference search is implemented as a state machine, so interaction is possible during execution.

**Related commands:** WAIT

**Mnemonic:** RFS <START|STOP|STATUS>, o

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
13	0 START – start ref. search 1 STOP – abort ref. search 2 STATUS – get status	o*	(don't care)

\*motor number is always 0 if only one motor is involved

**Reply in direct mode:**

When using type 0 (START) or 1 (STOP):

STATUS	VALUE
100 – OK	(don't care)

When using type 2 (STATUS):

STATUS	VALUE
100 – OK	0 – no ref. search active other values – ref. search is active

**Example:**

start reference search of motor #0

*Mnemonic:* RFS START, o

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
Value (hex)	\$01	\$0d	\$00	\$00	\$00	\$00	\$00	\$00	\$0f

### 5.7.14 SIO (set output)

This command sets the status of the general digital output either to low (0) or to high (1). On the TMCM-142 module three general purpose outputs are available, so the port number can be 0, 1 or 2.

**Internal function:** The passed value is transferred to the specified output line.

**Related commands:** GIO, WAIT

**Mnemonic:** SIO <port number>, <bank number>, <value>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
14	<port number>	<bank number>	<value>

**Reply structure:**

STATUS	VALUE
100 - OK	(don't care)

**Example:**

Set OUT\_1 to high (bank 2, output 1; general purpose output)

*Mnemonic:* SIO 1, 2, 1

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-Address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$0e	\$01	\$02	\$00	\$00	\$00	\$01	\$13

#### Overview of available output ports:

##### I/O bank 2 - digital outputs

Type	I/O line	Range
0	OUT1	0/1
1	OUT2	0/1
2	OUT3	0/1

### 5.7.15 GIO (get input/output)

With this command the status of the two available general purpose inputs of the module can be read out. The function reads a digital or analogue input port. Digital lines will read 0 and 1, while the ADC channels deliver their 10 bit result in the range of 0...1023. In stand-alone mode the requested value is copied to the "accumulator" (accu) for further processing purposes such as conditioned jumps. In direct mode the value is only output in the "value" field of the reply, without affecting the accumulator. The actual status of a digital output line can also be read.

**Internal function:** The specified line is read.

**Related commands:** SIO, WAIT

**Mnemonic:** GIO <port number>, <bank number>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
15	<port number>	<bank number>	(don't care)

**Reply in direct mode:**

STATUS	VALUE
100 - OK	<status of the port>

**Example:**

get the analogue value of ADC channel 3  
*Mnemonic:* GIO 3, 1

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$0f	\$03	\$01	\$00	\$00	\$00	\$00	\$14

*Reply:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Host-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$02	\$01	\$64	\$0f	\$00	\$00	\$01	\$fa	\$72

⇒ value: 506

#### Overview of available input ports:

##### I/O bank 0 - digital inputs

The ADIN lines can be read as digital or analogue inputs at the same time. The digital states can be accessed in bank 0. **When using a digital input it is often useful to switch on its pull-up resistor. Please see the SIO command on how to do this.**

Type	I/O line	Range
0	ADIN <sub>1</sub>	0/1
1	ADIN <sub>2</sub>	0/1
2	ADIN <sub>3</sub>	0/1
3	IN <sub>4</sub>	0/1

4	IN5	0/1
5	IN6	0/1
6	IN7	0/1
7	IN8	0/1
8	IN9	0/1

### I/O bank 1 – analogue inputs

The ADIN lines can be read as digital or analogue inputs at the same time. The analogue values can be accessed in bank 1. **When using an input in analogue mode its pull-up resistor must be switched off. Please see the SIO command how to do this (5.7.14).**

Type	I/O line	Range
0	ADIN1	0...1023
1	ADIN2	0...1023
2	ADIN3	0..1023

### I/O bank 2 – status information

Here, the status of the digital outputs (see section 5.7.14) can be read back.

### 5.7.16 CALC (calculate)

A value in the accumulator variable, previously read by a function such as GAP (get axis parameter), can be modified with this instruction. Nine different arithmetic functions can be chosen and one constant operand value must be specified. The result is written back to the accumulator, for further processing like comparisons or data transfer.

**Related commands:** CALCX, COMP, JC, AAP, AGP, GAP, GGP, GIO

**Mnemonic:** CALC <op>, <value>

where <op> is ADD, SUB, MUL, DIV, MOD, AND, OR, XOR, NOT or LOAD

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
19	0 ADD – add to accu 1 SUB – subtract from accu 2 MUL – multiply accu by 3 DIV – divide accu by 4 MOD – modulo divide by 5 AND – logical and accu with 6 OR – logical or accu with 7 XOR – logical exor accu with 8 NOT – logical invert accu 9 LOAD – load operand to accu	(don't care)	<operand>

**Example:**

multiply accu by -5000

*Mnemonic:* CALC MUL, -5000

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$13	\$02	\$00	\$FF	\$FF	\$EC	\$78	\$78

### 5.7.17 COMP (compare)

The specified number is compared to the value in the accumulator register. The result of the comparison can for example be used by the conditional jump (JC) instruction. This command is intended for use in stand-alone operation only.

**Note that the host address and the reply is only used to transfer this instruction to the TMCL™ program memory during downloading of a TMCL™ program. It does not make sense to use this command in direct mode.**

**Internal function:** The specified value is compared to the internal "accumulator", which holds the value of a preceding "get" or calculate instruction (see GAP/GGP/GIO/CALC/CALCX). The internal arithmetic status flags are set according to the comparison result.

**Related commands:** JC (jump conditional), GAP, GGP,GIO, CALC, CALCX

**Mnemonic:** COMP <value>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
20	(don't care)	(don't care)	<comparison value>

**Example:**

Jump to the address given by the label when the position of motor is greater than or equal to 1000.

```
GAP 1, 2, 0 //get axis parameter, type: no. 1 (actual position), motor: 0, value:0 (don't care)
COMP 1000 //compare actual value to 1000
JC GE, Label //jump, type: 5 greater/equal, the label must be defined somewhere else in the
program
```

*Binary format of the COMP 1000 command:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$14	\$00	\$00	\$00	\$00	\$03	\$e8	\$00

### 5.7.18 JC (jump conditional)

The JC instruction enables a conditional jump to a fixed address in the TMCL™ program memory, if the specified condition is met. The conditions refer to the result of a preceding comparison. Please refer to COMP instruction for examples. This function is for stand-alone operation only.

**Note that the host address and the reply is only used to transfer this instruction to the TMCL™ program memory. See the host-only control functions for details. It is not possible to use this command in direct mode.**

**Internal function:** the TMCL™ program counter is set to the passed value if the arithmetic status flags are in the appropriate state(s).

**Related commands:** JA, COMP, WAIT, CLE

**Mnemonic:** JC <condition>, <label>  
 where <condition>=ZE|NZ|EQ|NE|GT|GE|LT|LE|ETO|EAL|EDV|EPO

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
21	0 ZE - zero 1 NZ - not zero 2 EQ - equal 3 NE - not equal 4 GT - greater 5 GE - greater/equal 6 LT - lower 7 LE - lower/equal 8 ETO - time out error 9 EAL - external alarm 12 ESD - shutdown error	(don't care)	<jump address>

**Example:**

Jump to address given by the label when the position of motor is greater than or equal to 1000.

```
GAP 1, 0, 0 //get axis parameter, type: no. 1 (actual position), motor: 0, value: 0 (don't care)
COMP 1000 //compare actual value to 1000
JC GE, Label //jump, type: 5 greater/equal
...
...
Label: ROL 0, 1000
```

*Binary format of "JC GE, Label" when Label is at address 10:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
Value (hex)	\$01	\$15	\$05	\$00	\$00	\$00	\$00	\$0a	\$25

### 5.7.19 JA (jump always)

Jump to a fixed address in the TMCL™ program memory. This command is intended for stand-alone operation only.

**Note that the host address and the reply is only used to transfer this instruction to the TMCL™ program memory. This command cannot be used in direct mode.**

**Internal function:** the TMCL™ program counter is set to the passed value.

**Related commands:** JC, WAIT, CSUB

**Mnemonic:** JA <Label>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
22	(don't care)	(don't care)	<jump address>

**Example:** An infinite loop in TMCL™

```

Loop: MVP ABS, 0, 10000
      WAIT POS, 0, 0
      MVP ABS, 0, 0
      WAIT POS, 0, 0
      JA Loop      //Jump to the label "Loop"
    
```

*Binary format of "JA Loop" assuming that the label "Loop" is at address 20:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$16	\$00	\$00	\$00	\$00	\$00	\$14	\$2b

## 5.7.20 CSUB (call subroutine)

This function calls a subroutine in the TMCL™ program memory. It is intended for stand-alone operation only.

**Note that the host address and the reply is only used to transfer this instruction to the TMCL™ program memory. This command cannot be used in direct mode.**

**Internal function:** The actual TMCL™ program counter value is saved to an internal stack, afterwards overwritten with the passed value. The number of entries in the internal stack is limited to 8. This also limits nesting of subroutine calls to 8. The command will be ignored if there is no more stack space left.

**Related commands:** RSUB, JA

**Mnemonic:** CSUB <Label>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
23	(don't care)	(don't care)	<subroutine address>

**Example: Call a subroutine**

```

Loop: MVP ABS, 0, 10000
      CSUB SubW //Save program counter and jump to label "SubW"
      MVP ABS, 0, 0
      JA Loop

```

```

SubW: WAIT POS, 0, 0
      WAIT TICKS, 0, 50
      RSUB //Continue with the command following the CSUB command

```

*Binary format of the "CSUB SubW" command assuming that the label "SubW" is at address 100:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
Value (hex)	\$01	\$17	\$00	\$00	\$00	\$00	\$00	\$64	\$7c

### 5.7.21 RSUB (return from subroutine)

Return from a subroutine to the command after the CSUB command. This command is intended for use in stand-alone mode only.

**Note that the host address and the reply is only used to transfer this instruction to the TMCL™ program memory. This command cannot be used in direct mode.**

**Internal function:** The TMCL program counter is set to the last value of the stack. The command will be ignored if the stack is empty.

**Related command:** CSUB

**Mnemonic:** RSUB

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
24	(don't care)	(don't care)	(don't care)

**Example:** please see the CSUB example (section Fehler! Verweisquelle konnte nicht gefunden werden.).

*Binary format of RSUB:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
Value (hex)	\$01	\$18	\$00	\$00	\$00	\$00	\$00	\$00	\$19

### 5.7.22 WAIT (wait for an event to occur)

This instruction interrupts the execution of the TMCL™ program until the specified condition is met. This command is intended for stand-alone operation only.

**Note that the host address and the reply is only used to transfer this instruction to the TMCL™ program memory. This command is not to be used in direct mode.**

There are five different wait conditions that can be used:

- TICKS: Wait until the number of timer ticks specified by the <ticks> parameter has been reached.
- POS: Wait until the target position of the motor specified by the <motor> parameter has been reached. An optional timeout value (0 for no timeout) must be specified by the <ticks> parameter.
- REFSW: Wait until the reference switch of the motor specified by the <motor> parameter has been triggered. An optional timeout value (0 for no timeout) must be specified by the <ticks> parameter.
- LIMSW: Wait until a limit switch of the motor specified by the <motor> parameter has been triggered. An optional timeout value (0 for no timeout) must be specified by the <ticks> parameter.
- RFS: Wait until the reference search of the motor specified by the <motor> field has been reached. An optional timeout value (0 for no timeout) must be specified by the <ticks> parameter.

The timeout flag (ETO) will be set after a timeout limit has been reached. You can then use a JC ETO command to check for such errors or clear the error using the CLE command.

**Internal function:** The TMCL™ program counter is held until the specified condition is met.

**Related commands:** JC, CLE

**Mnemonic:** WAIT <condition>, 0, <ticks>  
 where <condition> is TICKS|POS|REFSW|LIMSW|RFS

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
27	0 TICKS - timer ticks* <sup>1</sup>	(don't care)	<no. of ticks*>
	1 POS - target position reached	0* <sup>2</sup> 0..2 resp. 0..5	<no. of ticks* for timeout>, 0 for no timeout
	2 REFSW – reference switch	0 0..2 resp. 0..5	<no. of ticks* for timeout>, 0 for no timeout
	3 LIMSW – limit switch	0 0..2 resp. 0..5	<no. of ticks* for timeout>, 0 for no timeout
	4 RFS – reference search completed	0 0..2 resp. 0..5	<no. of ticks* for timeout>, 0 for no timeout

\*<sup>1</sup> one tick is 10 milliseconds (in standard firmware)  
 \*<sup>2</sup> motor number is always 0 if only one motor is involved

**Example:**

wait for motor #0 to reach its target position, without timeout  
 Mnemonic: WAIT POS, 0, 0

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
Value (hex)	\$01	\$1b	\$01	\$00	\$00	\$00	\$00	\$00	\$1e

### 5.7.23 STOP (stop TMCL™ program execution)

This function stops executing a TMCL™ program. The host address and the reply are only used to transfer the instruction to the TMCL™ program memory.

***This command should be placed at the end of every stand-alone TMCL™ program. It is not to be used in direct mode.***

**Internal function:** TMCL™ instruction fetching is stopped.

**Related commands:** none

**Mnemonic:** STOP

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
28	(don't care)	(don't care)	(don't care)

**Example:**

*Mnemonic:* STOP

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$1c	\$00	\$00	\$00	\$00	\$00	\$00	\$1d

### 5.7.24 SCO (set coordinate)

Up to 20 position values (coordinates) can be stored for every axis for use with the MVP COORD command. This command sets a coordinate to a specified value. **Please note that the coordinate number 0 is only stored in RAM, all others are also stored in the EEPROM.**

**Internal function:** The passed value is stored in the internal position array.

**Related commands:** GCO, CCO, MVP

**Mnemonic:** SCO <coordinate number>, 0, <position>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
30	<coordinate number> (0...56)	0*	<position> (-2 <sup>23</sup> ...+2 <sup>23</sup> )

\* motor number

**Reply in direct mode:**

STATUS	VALUE
100 - OK	(don't care)

**Example:**

set coordinate #1 of motor to 1000

*Mnemonic:* SCO 1, 0, 1000

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
Value (hex)	\$01	\$1e	\$01	\$00	\$00	\$00	\$03	\$e8	\$0d

### 5.7.25 GCO (get coordinate)

This command makes possible to read out a previously stored coordinate. In stand-alone mode the requested value is copied to the accumulator register for further processing purposes such as conditioned jumps. In direct mode, the value is only output in the value field of the reply, without affecting the accumulator. **Please note that the coordinate number 0 is stored in RAM only, all others are also stored in the EEPROM.**

**Internal function:** The desired value is read out of the internal coordinate array, copied to the accumulator register and -in direct mode- returned in the "value" field of the reply.

**Related commands:** SCO, CCO, MVP

**Mnemonic:** GCO <coordinate number>, 0

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
31	<coordinate number> (0...56)	0*	(don't care)

\* motor number

**Reply in direct mode:**

STATUS	VALUE
100 - OK	(don't care)

**Example:**

get motor #0 value of coordinate 1  
*Mnemonic:* GCO 1, 0

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$1f	\$01	\$00	\$00	\$00	\$00	\$00	\$23

*Reply:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$02	\$01	\$64	\$0a	\$00	\$00	\$00	\$00	\$86

⇒ Value: 0

## 5.7.26 CCO (capture coordinate)

The actual position of the axe is copied to the selected coordinate variable. **Note that the coordinate number 0 is stored in RAM only and all others are also stored in the EEPROM.**

**Internal function:** The selected (24 bit) position values are written to the 20 by 3 bytes wide coordinate array.

**Related commands:** SCO, GCO, MVP

**Mnemonic:** CCO <coordinate number>, 0

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
32	<coordinate number> (0...56)	0*	(don't care)

\* motor number

**Reply in direct mode:**

STATUS	VALUE
100 – OK	(don't care)

**Example:**

store current position of the axe to coordinate 3

*Mnemonic:* CCO 3, 0

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
Value (hex)	\$01	\$20	\$03	\$00	\$00	\$00	\$00	\$00	\$2b

### 5.7.27 CALCX (calculate using the X register)

This instruction is very similar to CALC, but the second operand comes from the X register. The X register can be loaded with the LOAD or the SWAP type of this instruction. The result is written back to the accumulator for further processing like comparisons or data transfer.

**Related commands:** CALC, COMP, JC, AAP, AGP

**Mnemonic:** CALCX <operation>  
with <operation>=ADD|SUB|MUL|DIV|MOD|AND|OR|XOR|NOT|LOAD|SWAP

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
33	0 ADD – add X register to accu 1 SUB – subtract X register from accu 2 MUL – multiply accu by X register 3 DIV – divide accu by X-register 4 MOD – modulo divide accu by x-register 5 AND – logical and accu with X-register 6 OR – logical or accu with X-register 7 XOR – logical exor accu with X-register 8 NOT – logical invert X-register 9 LOAD – load accu to X-register 10 SWAP – swap accu with X-register	(don't care)	(don't care)

**Example:**

multiply accu by X-register  
Mnemonic: CALCX MUL

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$21	\$02	\$00	\$00	\$00	\$00	\$00	\$24

### 5.7.28 AAP (accumulator to axis parameter)

The content of the accumulator register is transferred to the specified axis parameter. For practical usage, the accumulator has to be loaded e.g. by a preceding GAP instruction. The accumulator may have been modified by the CALC or CALCX (calculate) instruction. (See chapter XXX for a complete list of axis parameters.)

**Related commands:** AGP, SAP, GAP, SGP, GGP, GIO, GCO, CALC, CALCX

**Mnemonic:** AAP <parameter number>, 0

#### Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
34	<parameter number>	0*	<don't care>

\* motor number

#### Reply in direct mode:

STATUS	VALUE
100 – OK	(don't care)

#### List of basic parameters, which can be used for AAP:

##### Meaning of the letters in column "Access":

R = readable (GAP)

W = writable (SAP)

E = automatically restored from EEPROM after reset or power-on

**Please note, that for the binary representation <parameter number> has to be filled with the number and the <value> has to be filled with a value from range.**

Number	Axis parameter	Description	Range	Access
0	Target position		-2147483648 .. +2147483647	RW
1	Actual position		-2147483648 .. +2147483647	RW
2	Target speed		-2147483648 .. +2147483647	RW
3	Actual speed		-2147483648 .. +2147483647	RW
4	Max. positioning speed	Speed used for positioning (MVP commands).	0..2147483647	RWE
5	Max. acceleration and deceleration	Sets acceleration and deceleration to the same value.	1..16777215	RWE
6	Max. current	Current when motor is running. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
7	Standby current	Current when motor is standing. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
12	Right limit switch disable	Deactivates the function of the right limit switch when set to 1.	0/1	RWE
13	Left limit switch disable	Deactivates the function of the left limit switch when set to 1.	0/1	RWE
14	Switch mode			RWE
15	Stop deceleration	Deceleration when touching a stop switch.	1..16777215	RWE

16	Max. acceleration	Acceleration	1..16777215	RWE
17	Max. deceleration	Deceleration	1..16777215	RWE
18	Bow	0: trapezoidal ramps 1..18: S-shaped ramps	0..18	RWE
19	Shaft	Reverses the motor direction when set to 1.	0/1	RWE
20	Standby delay		0..4095	
21	Mixed decay run	0: no mixed decay when running 1: use mixed decay when running	0/1	RWE
22	Mixed decay standby	0: no mixed decay when standing 1: use mixed decay when standing	0/1	RWE
23	Chopper clock divider		96..818	RWE
24	StallGuard threshold	The motor will be stopped when the actual load value exceeds this threshold value.	0..7	RWE
27	Microstep resolution	0: 2048 micro steps 1: 1024 micro steps 2: 512 micro steps 3: 256 micro steps 4: 128 micro steps 5: 64 micro steps 6: 32 micro steps 7: 16 micro steps 8: 8 micro steps 9: 4 micro steps 10: 2 micro steps 11: 1 full step	0..11	RWE
28	PID tolerance			RWE
29	Sine wave offset		0..255	RWE
30	PID p factor		0..1677215	RWE
31	PID i factor		0..1677215	RWE
32	PID d factor		0..1677215	RWE
33	PID i clipping		0..32640	RWE
35	PID d clock divider		0..255	RWE
36	PID dv correction		-2147483648 .. +2147483647	RW
37	PID dv clipping		0..2147483647	RWE
40	PID mode	0: do not use PID 1: use PID	0/1	RWE
42	Encoder constant	With every pulse this constant is added to or subtracted from the encoder position register.	0..2147483647	RWE
43	Encoder clear mode			RWE
47	Encoder warn distance	Maximum deviation between motor and encoder.		RWE
48	Compare position	Tbd	-2147483648 .. +2147483647	RWE
50	Right position limit	Position limit when moving in positive direction.	-2147483648 .. +2147483647	RWE
51	Left position limit	Position limit when moving in negative direction.	-2147483648 .. +2147483647	RWE
52	Right position limit enable	The motor cannot drive beyond the right position limit when set to 1.	0/1	RWE
53	Left position limit enable	The motor cannot drive beyond the left position limit when set to 1.	0/1	RWE
63	Microstep table position		0..8191	RW

64	Step pulse length	Length of the step pulses on the step/direction output.	0..255	RWE
128	Ramp mode	Normally set automatically by the ROL, ROR, MVP and MST commands. 0: positioning mode 1: reserved 2: velocity mode 3: hold mode	0..3	RW
193	Reference search mode			RWE
194	Reference search speed			RWE
195	Reference switch speed			RWE
255	Unit conversion mode	Units to use for velocity and acceleration: 0: use TMC457 units 1: use PPS units	0/1	RWE

**Example:**

Positioning motor by a potentiometer connected to the analogue input #0:

```

Start:  GIO 0,1      // get value of analogue input line 0
        CALC MUL, 4 // multiply by 4
        AAP 0,0     // transfer result to target position of motor 0
        JA Start    // jump back to start
    
```

*Binary format of the AAP 0,0 command:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$22	\$00	\$00	\$00	\$00	\$00	\$00	\$23

### 5.7.29 AGP (accumulator to global parameter)

The content of the accumulator register is transferred to the specified global parameter. For practical usage, the accumulator has to be loaded e.g. by a preceding GAP instruction. The accumulator may have been modified by the CALC or CALCX (calculate) instruction. **Note that the global parameters in bank 0 are EEPROM-only and thus should not be modified automatically by a stand-alone application.** (See chapter 7 for a complete list of global parameters).

**Related commands:** AAP, SGP, GGP, SAP, GAP, GIO

**Mnemonic:** AGP <parameter number>, <bank number>

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
35	<parameter number>	<bank number>	(don't care)

**Reply in direct mode:**

STATUS	VALUE
100 – OK	(don't care)

**Global parameters of bank 0, which can be used for AGP:**

**Meaning of the letters in column "Access":**

- R = readable (GGP)
- W = writeable (SGP)
- E = automatically restored from EEPROM after reset or power-on.

Number	Global parameter	Description	Range	Access
64	EEPROM magic	Setting this parameter to a different value as \$E4 will cause re-initialization of the axis and global parameters (to factory defaults) after the next power up. This is useful in case of miss-configuration.	0...255	RWE
65	RS232/RS485 baud rate	0 – 9600 baud (default) 1 – 14400 baud 2 – 19200 baud 3 – 28800 baud 4 – 38400 baud 5 – 57600 baud 6 – 76800 baud <b>Not supported by Windows!</b> 7 – 115200 baud <b>115200 does not work with most host PCs as the baud rate error on the module is too high (-3.5% baud rate error).</b>	0...7	RWE
66	Serial address	The module (target) address for RS-232/RS-485.	0...255	RWE
67	ASCII mode	Configure the TMCL™ ASCII interface: Bit 0: 0 – start up in binary (normal) mode 1 – start up in ASCII mode Bits 4 and 5: 00 – Echo back each character 01 – Echo back complete command 10 – Do not send echo, only send command reply		RWE

Number	Global parameter	Description	Range	Access
69	CAN bit rate	1 – 10kBit/s 2 – 20kBit/s 3 – 50kBit/s 4 – 100kBit/s 5 – 125kBit/s 6 – 250kBit/s (default) 7 – 500kBit/s 8 – 1000kBit/s	1..7	RWE
70	CAN reply ID	The CAN ID for replies from the board (default: 2)	0..7ff	RWE
71	CAN ID	The module (target) address for CAN (default: 1)	0..7ff	RWE
73	Configuration EEPROM lock flag	Write: 1234 to lock the EEPROM, 4321 to unlock it. Read: 1=EEPROM locked, 0=EEPROM unlocked.	0/1	RWE
75	Telegram pause time	Pause time before the reply via RS232 or RS485 is sent. For RS232 set to 0. For RS485 it is often necessary to set it to 15 (for RS485 adapters controlled by the RTS pin). For CAN interface this parameter has no effect!	0...255	RWE
76	Serial host address	Host address used in the reply telegrams sent back via RS232 or RS485.	0..255	RWE
77	Auto start mode	0: Do not start TMCL™ application after power up (default). 1: Start TMCL™ application automatically after power up.	0/1	RWE
80	Shutdown pin functionality	Select the functionality of the SHUTDOWN pin 0 – no function 1 – high active 2 – low active	0..2	RWE
81	TMCL™ code protection	Protect a TMCL™ program against disassembling or overwriting. 0 – no protection 1 – protection against disassembling 2 – protection against overwriting 3 – protection against disassembling and overwriting <b><i>If you switch off the protection against disassembling, the program will be erased first! Changing this value from 1 or 3 to 0 or 2, the TMCL™ program will be wiped off.</i></b>	0,1,2,3	RWE
132	Tick timer	A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value.		RW

#### **Global parameters of bank 1, which can be used for AGP:**

The global parameter bank 1 can be used in customer specific extensions of the firmware. Such extensions will be done by Trinamic on customer request.

#### **Global parameters of bank 2, which can be used for AGP:**

Bank 2 contains general purpose 32 bit variables for the use in TMCL™ applications. They are located in RAM and can be stored to EEPROM. After booting, their values are automatically restored to the RAM.

Number	Global parameter	Description	Range	Access
0	General purpose variable #0	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
1	General purpose variable #1	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
2	General purpose variable #2	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
3	General purpose variable #3	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE

Number	Global parameter	Description	Range	Access
4	General purpose variable #4	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
5	General purpose variable #5	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
6	General purpose variable #6	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
7	General purpose variable #7	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
8	General purpose variable #8	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
9	General purpose variable #9	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
10	General purpose variable #10	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
11	General purpose variable #11	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
12	General purpose variable #12	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
13	General purpose variable #13	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
14	General purpose variable #14	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
15	General purpose variable #15	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
16	General purpose variable #16	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
17	General purpose variable #17	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
18	General purpose variable #18	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
19	General purpose variable #19	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE
20..55	General purpose variables #20..#55	for use in TMCL™ applications	-2 <sup>31</sup> ...+2 <sup>31</sup>	RWE

*Please refer to chapter 7 for more information about bank 0 to 2.*

**Example:**

copy accumulator to TMCL™ user variable #3  
*Mnemonic: AGP 3, 2*

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	\$01	\$23	\$03	\$02	\$00	\$00	\$00	\$00	\$29

### 5.7.30 CLE (clear error flags)

This command clears the internal error flags. *It is intended for use in stand-alone mode only and must not be used in direct mode.*

The following error flags can be cleared by this command (determined by the <flag> parameter):

- ALL: clear all error flags.
- ETO: clear the timeout flag.
- EAL: clear the external alarm flag
- EDV: clear the deviation flag (modules with encoder feedback only, e.g. TMCM-100)
- EPO: clear the position error flag (modules with encoder feedback only, e.g. TMCM-100)

**Related commands:** JC

**Mnemonic:** CLE <flags>

where <flags>=ALL|ETO|EDV|EPO

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
36	0 - (ALL) all flags 1 - (ETO) timeout flag 2 - (EAL) alarm flag 3 - (EDV) deviation flag 4 - (EPO) position flag 5 - (ESD) shutdown flag	(don't care)	(don't care)

**Example:**

Reset the timeout flag

*Mnemonic:* CLE ETO

*Binary:*

Byte Index	0	1	2	3	4	5	6	7	8
<b>Function</b>	Target-address	Instruction Number	Type	Motor/Bank	Operand Byte <sub>3</sub>	Operand Byte <sub>2</sub>	Operand Byte <sub>1</sub>	Operand Byte <sub>0</sub>	Checksum
<b>Value (hex)</b>	\$01	\$24	\$01	\$00	\$00	\$00	\$00	\$00	\$26

### 5.7.31 Customer specific TMCL™ command extension (UFo...UF7/user function)

The user definable functions UFo through UF7 are predefined, "empty" functions for user specific purposes. Contact TRINAMIC for customer specific programming of these functions.

**Internal function:** Call user specific functions implemented in "C" by TRINAMIC.

**Related commands:** none

**Mnemonic:** UFo .. UF7

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
64...71	(user defined)	(user defined)	(user defined)

**Reply in direct mode:**

Byte Index	0	1	2	3	4	5	6	7	8
<b>Function</b>	Target-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
<b>Value (hex)</b>	\$02	\$01	(user defined)	64...71	(user defined)	(user defined)	(user defined)	(user defined)	<checksum >

### 5.7.32 Request target position reached event

This command is the only exception to the TMCL™ protocol, as it sends two replies: One immediately after the command has been executed (like all other commands also), and one additional reply that will be sent when the motor has reached its target position. ***This instruction can only be used in direct mode (in stand-alone mode, it is covered by the WAIT command) and hence does not have a mnemonic.***

**Internal function:** Send an additional reply when the motor has reached its target position

**Mnemonic:** ---

**Binary representation:**

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
138	(don't care)	(don't care)	o*

\* motor number

**Reply in direct mode (right after execution of this command):**

Byte Index	0	1	2	3	4	5	6	7	8
<b>Function</b>	Target-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
<b>Value (hex)</b>	\$02	\$01	100	138	\$00	\$00	\$00	Motor bit mask	<checksum >

**Additional reply in direct mode (after motors have reached their target positions):**

Byte Index	0	1	2	3	4	5	6	7	8
<b>Function</b>	Target-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
<b>Value (hex)</b>	\$02	\$01	128	138	\$00	\$00	\$00	Motor bit mask	<checksum >

### 5.7.33 BIN (return to binary mode)

This command can only be used in ASCII mode. It quits the ASCII mode and returns to binary mode.

**Related Commands:** none

**Mnemonic:** BIN

**Binary representation:** This command does not have a binary representation as it can only be used in ASCII mode.

## 5.7.34 TMCL™ Control Functions

*The following functions are for host control purposes only and are not allowed for stand-alone mode. In most cases, there is no need for the customer to use one of those functions (except command 139). They are mentioned here only for reasons of completeness. These commands have no mnemonics, as they cannot be used in TMCL™ programs. The Functions are to be used only by the TMCL-IDE to communicate with the module, for example to download a TMCL™ application into the module.*

**The only control commands that could be useful for a user host application are:**

- "get firmware revision" (command 136, please note the special reply format of this command, described at the end of this section)
- 129 (run application)

*All other functions can be achieved by using the appropriate functions of the TMCL-IDE.*

Instruction	Description	Type	Mot/Bank	Value
128 – stop application	a running TMCL™ standalone application is stopped	(don't care)	(don't care)	(don't care)
129 – run application	TMCL™ execution is started (or continued)	0 - run from current address 1 - run from specified address	(don't care)	(don't care) starting address
130 – step application	only the next command of a TMCL™ application is executed	(don't care)	(don't care)	(don't care)
131 – reset application	the program counter is set to zero, and the standalone application is stopped (when running or stepped)	(don't care)	(don't care)	(don't care)
132 – start download mode	target command execution is stopped and all following commands are transferred to the TMCL™ memory	(don't care)	(don't care)	starting address of the application
133 – quit download mode	target command execution is resumed	(don't care)	(don't care)	(don't care)
134 – read TMCL™ memory	the specified program memory location is read	(don't care)	(don't care)	<memory address>
135 – get application status	one of these values is returned: 0 – stop 1 – run 2 – step 3 – reset	(don't care)	(don't care)	(don't care)
136 – get firmware version	return the module type and firmware revision either as a string or in binary format	0 – string 1 – binary	(don't care)	(don't care)
137 – restore factory settings	reset all settings stored in the EEPROM to their factory defaults This command does not send back a reply.	(don't care)	(don't care)	must be 1234
138 – reserved				
139 – enter ASCII mode	Enter ASCII command line (see chapter 5.6)	(don't care)	(don't care)	(don't care)

**Special reply format of command 136:****Type set to 0 - reply as a string:**

Byte index	Contents
1	Host Address
2..9	Version string (8 characters, e.g. 140V2.50)

- There is no checksum in this reply format!
- To get also the last byte when using the CAN bus interface, just send this command in an eight byte frame instead of a seven byte frame. Then, eight bytes will be sent back, so you will get all characters of the version string.

**Type set to 1 - version number in binary format:**

- Please use the normal reply format.
- The version number is output in the "value" field of the reply in the following way:

Byte index in value field	Contents
1	Version number, low byte
2	Version number, high byte
3	Type number, low byte (currently not used)
4	Type number, high byte (currently not used)

## 6 Axis parameters

The following section describes all axis parameters that can be used with the SAP, GAP, AAP, STAP and RSAP commands.

### Meaning of the letters in column "Access":

R = readable (GAP)

W = writable (SAP)

E = automatically restored from EEPROM after reset or power-on

Number	Axis parameter	Description	Range	Access
0	Target position		-2147483648 .. +2147483647	RW
1	Actual position		-2147483648 .. +2147483647	RW
2	Target speed		-2147483648 .. +2147483647	RW
3	Actual speed		-2147483648 .. +2147483647	RW
4	Max. positioning speed	Speed used for positioning (MVP commands).	0..2147483647	RWE
5	Max. acceleration and deceleration	Sets acceleration and deceleration to the same value.	1..16777215	RWE
6	Max. current	Current when motor is running. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
7	Standby current	Current when motor is standing. 0 means 0%, 15 means 100% of the maximum possible current.	0..15	RWE
8	Position reached	1 when the target position and the actual position are equal.	0/1	R
9	Switch status			R
10	Right limit switch status	Logic state of the right switch.	0/1	R
11	Left limit switch status	Logic state of the left switch.	0/1	R
12	Right limit switch disable	Deactivates the function of the right limit switch when set to 1.	0/1	RWE
13	Left limit switch disable	Deactivates the function of the left limit switch when set to 1.	0/1	RWE
14	Switch mode			RWE
15	Stop deceleration	Deceleration when touching a stop switch.	1..16777215	RWE
16	Max. acceleration	Acceleration	1..16777215	RWE
17	Max. deceleration	Deceleration	1..16777215	RWE
18	Bow	0: trapezoidal ramps 1..18: S-shaped ramps	0..18	RWE
19	Shaft	Reverses the motor direction when set to 1.	0/1	RWE
20	Standby delay		0..4095	RWE
21	Mixed decay run	0: no mixed decay when running 1: use mixed decay when running	0/1	RWE
22	Mixed decay standby	0: no mixed decay when standing 1: use mixed decay when standing	0/1	RWE
23	Chopper clock divider		96..818	RWE
24	StallGuard threshold	The motor will be stopped when the actual load value exceeds this threshold value.	0..7	RWE
25	Actual load value		0..7	R

26	Driver status	Bit 0: driver error Bit 1: over temperature pre-warning Bit 2: motor stall (StallGuard)		R
27	Microstep resolution	0: 2048 micro steps 1: 1024 micro steps 2: 512 micro steps 3: 256 micro steps 4: 128 micro steps 5: 64 micro steps 6: 32 micro steps 7: 16 micro steps 8: 8 micro steps 9: 4 micro steps 10: 2 micro steps 11: 1 full step	0..11	RWE
28	PID tolerance			RWE
29	Sine wave offset		0..255	RWE
30	PID p factor		0..1677215	RWE
31	PID i factor		0..1677215	RWE
32	PID d factor		0..1677215	RWE
33	PID i clipping		0..32640	RWE
34	PID i sum			R
35	PID d clock divider		0..255	RWE
36	PID dv correction		-2147483648 .. +2147483647	RW
37	PID dv clipping		0..2147483647	RWE
38	PID error			R
39	PID Vactual			R
40	PID mode	0: do not use PID 1: use PID	0/1	RWE
41	Encoder position	Actual position of the encoder.	-2147483648 .. +2147483647	R
42	Encoder constant	With every pulse this constant is added to or subtracted from the encoder position register.	0..2147483647	RWE
43	Encoder clear mode			RWE
44	Encoder status	1 when an encoder null channel event has been detected. Cleared after reading.	0/1	R
45	Encoder latch	Encoder position latched on N channel event.		R
46	Position latch	Motor position latched on stop switch event.		R
47	Encoder warn distance	Maximum deviation between motor and encoder.		RWE
48	Compare position	Tbd	-2147483648 .. +2147483647	RWE
50	Right position limit	Position limit when moving in positive direction.	-2147483648 .. +2147483647	RWE
51	Left position limit	Position limit when moving in negative direction.	-2147483648 .. +2147483647	RWE
52	Right position limit enable	The motor cannot drive beyond the right position limit when set to 1.	0/1	RWE
53	Left position limit enable	The motor cannot drive beyond the left position limit when set to 1.	0/1	RWE
63	Microstep table position		0..8191	RW

64	Step pulse length	Length of the step pulses on the step/direction output.	0..255	RWE
128	Ramp mode	Normally set automatically by the ROL, ROR, MVP and MST commands. 0: positioning mode 1: reserved 2: velocity mode 3: hold mode	0..3	RW
129	Status flags	Bit 0: target position reached (same as parameter 8) Bit 1: target velocity reached (same as parameter 130) Bit 2: motor not moving (v=0) Bit 3: encoder warn distance exceeded		R
130	Velocity reached	Reads 1 when the actual speed is equal to the target speed	0/1	R
193	Reference search mode			RWE
194	Reference search speed			RWE
195	Reference switch speed			RWE
255	Unit conversion mode	Units to use for velocity and acceleration: 0: use TMC457 units 1: use PPS units	0/1	RWE

## 7 Global parameters

The global parameters apply for all types of TMC modules.

They are grouped into 3 banks:

- bank 0 (global configuration of the module)
- bank 1 (user C variables)
- bank 2 (user TMCL™ variables)

*Please use SGP and GGP commands to write and read global parameters (s. sections 5.7.10 and 5.7.9 ).*

### 7.1 Bank 0

Parameters with numbers from 64 on configure stuff like the serial address of the module RS232/RS485 baud rate or the CAN bit rate. Change these parameters to meet your needs. The best and easiest way to do this is to use the appropriate functions of the TMCL-IDE. The parameters with numbers between 64 and 128 are stored in EEPROM only.

*An SGP command on such a parameter will always store it permanently and no extra STGP command is needed.*

*Take care when changing these parameters, and use the appropriate functions of the TMCL-IDE to do it in an interactive way.*

Note: The TMC-142 does not have the parameters 0...38. They are used for modules which address more than one motor.

Meaning of the letters in column "Access":

- R = readable (GGP)
- W = writeable (SGP)
- E = automatically restored from EEPROM after reset or power-on.

Number	Global parameter	Description	Range	Access
64	EEPROM magic	Setting this parameter to a different value as \$E4 will cause re-initialization of the axis and global parameters (to factory defaults) after the next power up. This is useful in case of miss-configuration.	0...255	RWE
65	RS232/RS485 baud rate	0 – 9600 baud (default) 1 – 14400 baud 2 – 19200 baud 3 – 28800 baud 4 – 38400 baud 5 – 57600 baud 6 – 76800 baud <b>Not supported by Windows!</b> 7 – 115200 baud <b>115200 does not work with most host PCs as the baud rate error on the module is too high (-3.5% baud rate error).</b>	0...7	RWE
66	Serial address	The module (target) address for RS-232/RS-485.	0...255	RWE
67	ASCII mode	Configure the TMCL™ ASCII interface: Bit 0: 0 – start up in binary (normal) mode 1 – start up in ASCII mode Bits 4 and 5: 00 – Echo back each character 01 – Echo back complete command 10 – Do not send echo, only send command reply		RWE

Number	Global parameter	Description	Range	Access
69	CAN bit rate	1 – 10kBit/s 2 – 20kBit/s 3 – 50kBit/s 4 – 100kBit/s 5 – 125kBit/s 6 – 250kBit/s (default) 7 – 500kBit/s 8 – 1000kBit/s	1..7	RWE
70	CAN reply ID	The CAN ID for replies from the board (default: 2)	0..7ff	RWE
71	CAN ID	The module (target) address for CAN (default: 1)	0..7ff	RWE
73	Configuration EEPROM lock flag	Write: 1234 to lock the EEPROM, 4321 to unlock it. Read: 1=EEPROM locked, 0=EEPROM unlocked.	0/1	RWE
75	Telegram pause time	Pause time before the reply via RS232 or RS485 is sent. For RS232 set to 0. For RS485 it is often necessary to set it to 15 (for RS485 adapters controlled by the RTS pin). For CAN interface this parameter has no effect!	0...255	RWE
76	Serial host address	Host address used in the reply telegrams sent back via RS232 or RS485.	0..255	RWE
77	Auto start mode	0: Do not start TMCL™ application after power up (default). 1: Start TMCL™ application automatically after power up.	0/1	RWE
80	Shutdown pin functionality	Select the functionality of the SHUTDOWN pin 0 – no function 1 – high active 2 – low active	0..2	RWE
81	TMCL™ code protection	Protect a TMCL™ program against disassembling or overwriting. 0 – no protection 1 – protection against disassembling 2 – protection against overwriting 3 – protection against disassembling and overwriting <b><i>If you switch off the protection against disassembling, the program will be erased first! Changing this value from 1 or 3 to 0 or 2, the TMCL™ program will be wiped off.</i></b>	0,1,2,3	RWE
128	TMCL™ application status	0 – stop 1 – run 2 – step 3 – reset	0..3	R
129	Download mode	0 – normal mode 1 – download mode	0/1	R
130	TMCL™ program counter	The index of the currently executed TMCL™ instruction.		R
132	Tick timer	A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value.		RW

## 7.2 Bank 1

The global parameter bank 1 can be used in customer specific extensions of the firmware (also together with the UFO...UF7 commands). Please contact Trinamic if you would like to have customer specific extensions of the firmware.

### 7.3 Bank 2

Bank 2 contains general purpose 32 bit variables for the use in TMCL™ applications. They are located in RAM and can be stored to EEPROM. After booting, their values are automatically restored to the RAM.

Up-to, 56 user variables are available.

**Meaning of the letters in column "Access":**

- R = readable (GGP)
- W = writeable (SGP)
- E = automatically restored from EEPROM after reset or power-on.

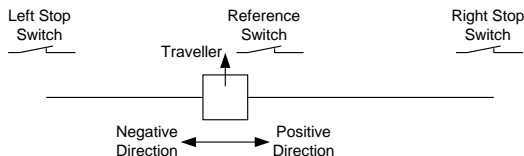
Number	Global parameter	Description	Range	Access
0	General purpose variable #0	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
1	General purpose variable #1	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
2	General purpose variable #2	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
3	General purpose variable #3	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
4	General purpose variable #4	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
5	General purpose variable #5	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
6	General purpose variable #6	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
7	General purpose variable #7	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
8	General purpose variable #8	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
9	General purpose variable #9	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
10	General purpose variable #10	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
11	General purpose variable #11	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
12	General purpose variable #12	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
13	General purpose variable #13	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
14	General purpose variable #14	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
15	General purpose variable #15	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
16	General purpose variable #16	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
17	General purpose variable #17	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
18	General purpose variable #18	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
19	General purpose variable #19	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE
20..55	General purpose variables #20..#55	for use in TMCL™ applications	$-2^{31} \dots +2^{31}$	RWE

## 8 Hints and tips

This chapter gives some hints and tips on using the functionality of TMCL™, for example how to use and parameterize the built-in reference point search algorithm or the incremental encoder interface.

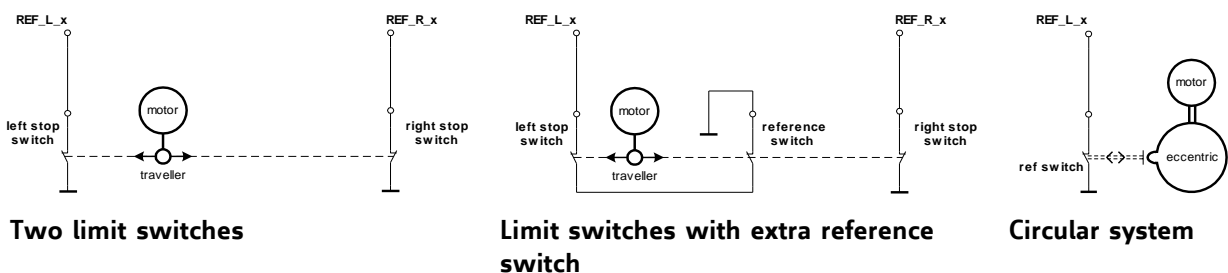
### 8.1 Reference search

The built-in reference search features switching point calibration and support of one or two reference switches. The internal operation is based on three individual state machines (one per axis) that can be started, stopped and monitored (instruction RFS, no. 13). The settings of the automatic stop functions corresponding to the switches (axis parameters 12 and 13) have no influence on the reference search.



#### Definition of the switches

- Selecting the referencing mode (axis parameter 193): in modes 1 and 2, the motor will start by moving "left" (negative position counts). In mode 3 (three-switch mode), the right stop switch is searched first to distinguish the left stop switch from the reference switch by the order of activation when moving left (reference switch and left limit switch share the same electrical function).
- Until the reference switch is found for the first time, the searching speed is identical to the maximum positioning speed (axis parameter 4), unless reduced by axis parameter 194.
- After hitting the reference switch, the motor slowly moves right until the switch is released. Finally the switch is re-entered in left direction, setting the reference point to the center of the two switching points. This low calibrating speed is a quarter of the maximum positioning speed by default (axis parameter 195).
- In TMC-142 hardware manual the connection of the left and the right limit switch is shown. The drawing in this section also shows the connection of three switches as left and right limit switch and a reference switch for the reference point. The reference switch is connected in series with the left limit switch. The differentiation between the left limit switch and the reference switch is made through software. Switches with open contacts (normally closed) are used.
- In circular systems there are no end points and thus only one reference switch is used for finding the reference point.



### 8.2 Fixing microstep errors

Due to the "zero crossing problem" of the TMC239 stepper motor drivers, microstep errors may occur with some motors as the minimum motor current that can be reached is slightly higher than zero (depending on the inductivity, resistance and supply voltage of the motor).

This can be solved by setting the "mixed decay run" parameter (axis parameter number 21) to the value 1. This switches on mixed decay when the motor is running. A further optimization is possible by changing the "sine wave offset" parameter (axis parameter 29). Please set the microstep resolution (parameter 27) again after changing this value to start the re-calculation of the microstep wave table. By changing the sine wave offset the wave can be adapted to the needs of a specific motor type.

**Use SAP 21, 0, 1 to turn on mixed decay and SAP 29, 0, x to change the sine wave offset.**

## 8.3 Using the RS485 interface

With most RS485 converters that can be attached to the COM port of a PC the data direction is controlled by the RTS pin of the COM port. Please note that this will only work with Windows 2000, Windows XP or Windows NT4, not with Windows 95, Windows 98 or Windows ME (due to a bug in these operating systems). Another problem is that Windows 2000/XP/NT4 switches the direction back to "receive" too late. To overcome this problem, set the "telegram pause time" (global parameter #75) of the module to 15 (or more if needed) by issuing an "SGP 75, 0, 15" command in direct mode. The parameter will automatically be stored in the configuration EEPROM.

***For RS232 set the "telegram pause time" to zero for maximum data throughput***

## 9 Revision history

### 9.1 Firmware revision

Version	Date	Description
4.17	2009-02-28	First version supporting all TMCL™ features

### 9.2 Document Revision

Version	Date	Author	Description
1.00	2009-02-28	SD	Initial version
1.01	2009-03-16	OK	Some corrections and extensions

## 10 References

[TMCL™] TMCL™ reference and programming manual (see <http://www.trinamic.com>)