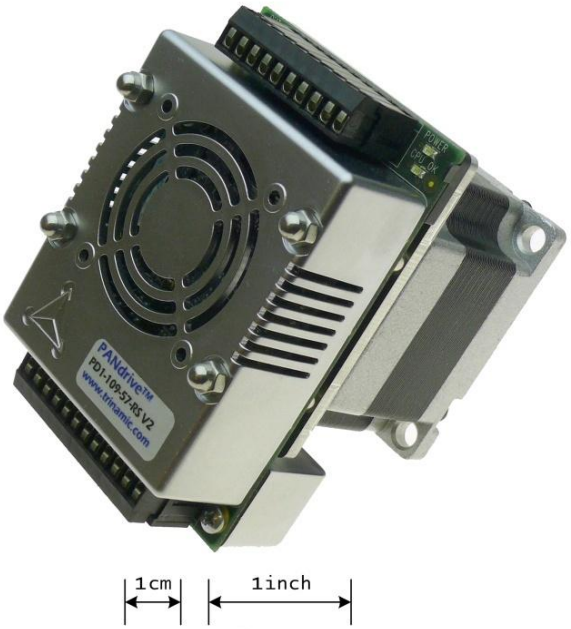
# PDx-109-57 V2




# TMCL™ Firmware Manual

Version: 1.00
2009-FEB-18

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

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.

# 2 Features

The PANdrive PD*x*-109-57 V2 features a full mechatronic solution including a 57mm flange motor. It is based on the TMCM-109-57 electronics and offers RS232 and RS485 interfaces. The power supply, the interface and the multipurpose I/Os can be connected via two pluggable screw terminal connectors. With the stallGuard™ feature it is possible to detect motor overload or motor stall.

The TMCM-109-57 comes with the PC based software development environment TMCL-IDE for the Trinamic Motion Control Language (TMCL™). Using predefined TMCL™ high level commands like *move to position* or *constant rotation* a rapid and fast development of motion control applications is guaranteed. Communication traffic is kept very low since all time critical operations, e.g. ramp calculation are performed onboard. The TMCL™ program can be stored in the on-board EEPROM for stand-alone operation. The firmware of the module can be updated via the serial interface.

**Applications**
- decentralized mechatronic drive with integrated intelligence
- high-precision drives with high dynamics and torque

**Electrical data**
- 18V to 55V motor supply voltage for highest motor dynamics
- up to 3.5A RMS nominal motor current

**Motor data**
- all PANdrive Motors optimized for 3.0A RMS motor current
- flange max. 56.5mm x 56.5mm
- D-cut of 15mm length and 0.5mm depth
- more specifications:

| Specifications | Parameter | Units | QSH5718 | | | |
|---|---|---|---|---|---|---|
| | | | -41-28-055 | -51-28-101 | 56-28-126 | -76-28-189 |
| Number of Leads | | N° | 4 | 4 | 4 | 4 |
| Step Angle | | ° | 1.8 | 1.8 | 1.8 | 1.8 |
| Step Angle Accuracy | | % | 5 | 5 | 5 | 5 |
| Rated Voltage | $V_{RATED}$ | V | 2 | 2.3 | 2.5 | 3.2 |
| Rated Phase Current | $I_{RMS\ RATED}$ | A | 2.8 | 2.8 | 2.8 | 2.8 |
| Phase Resistance at 20°C | $R_{COIL}$ | Ω | 0.7 | 0.83 | 0.9 | 1.13 |
| Phase Inductance (typ.) | | mH | 1.4 | 2.2 | 2.5 | 3.6 |
| Holding Torque | | Nm | 0.55 | 1.01 | 1.26 | 1.89 |
| Detent Torque | | Nm | 0.020 | 0.035 | 0.039 | 0.066 |
| Rotor Inertia | | g cm² | 120 | 275 | 300 | 480 |
| Insulation Class | | | B | B | B | B |
| Max. applicable voltage | | V | 75 | 75 | 75 | 75 |
| Max. radial force (20mm from front flange) | | N | 75 | 75 | 75 | 75 |
| Max. axial force | | N | 15 | 15 | 15 | 15 |
| Weight | | kg | 0.45 | 0.65 | 0.7 | 1 |
| Length | | mm | 41 | 51 | 56 | 76 |
| Temp. Rise (rated current, 2 phase on) | | °C | +80 max | +80 max | +80 max | +80 max |
| Ambient Temperature | | °C | -20 +50 | -20 +50 | -20 +50 | -20 +50 |

**Table 2.1: Specifications of the PANdrive motors**

**Interface**
- RS232, RS485
- 2 inputs for reference and stop switches
- 3 general purpose inputs and 1 general purpose output

**Features**
- up to 16 times microstepping
- memory for 2048 TMCL commands
- automatic ramp generation in hardware
- on the fly alteration of motion parameters (e.g. position, velocity, acceleration)
- stallGuard™ for sensorless motor stall detection
- optically isolated inputs for two general purpose inputs and the disable input
- dynamic current control

**Software**
- stand-alone operation using TMCL™ or remote controlled operation
- PC-based application development software TMCL-IDE included

**Other**
- Pluggable screw terminal connectors for all external signals
- RoHS compliant latest from July 1st, 2006

# 3 Order codes

| Order code | Description | Dimensions |
|---|---|---|
| PD1-109-57-RS | PANdrive 0.55Nm | 93.6 x 57.2 x 86 |
| PD2-109-57-RS | PANdrive 1.01Nm | 103.6 x 57.2 x 86 |
| PD3-109-57-RS | PANdrive 1.26Nm | 108.6 x 57.2 x 86 |
| PD4-109-57-RS | PANdrive 1.89Nm | 128.6 x 57.2 x 86 |
| **Option** | **Host interface** | |
| RS | RS232 and RS485 | |

**Table 3.1: Order codes**

# 4 Overview

As with most TRINAMIC modules the software running on the microprocessor of the PD-109-57 V2 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 [TMCL] shipped with most of TRINAMIC modules with regard to protocol and commands. Corresponding, the module is based on the TMC428-I stepper motor controller and the TMC249 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 Putting the PDx-109-57 V2 into operation

Here you can find basic information for putting your PANdrive™ into operation. The text contains a simple example for a TMCL™ program and a short description of operating the module in direct mode.

The things you need:

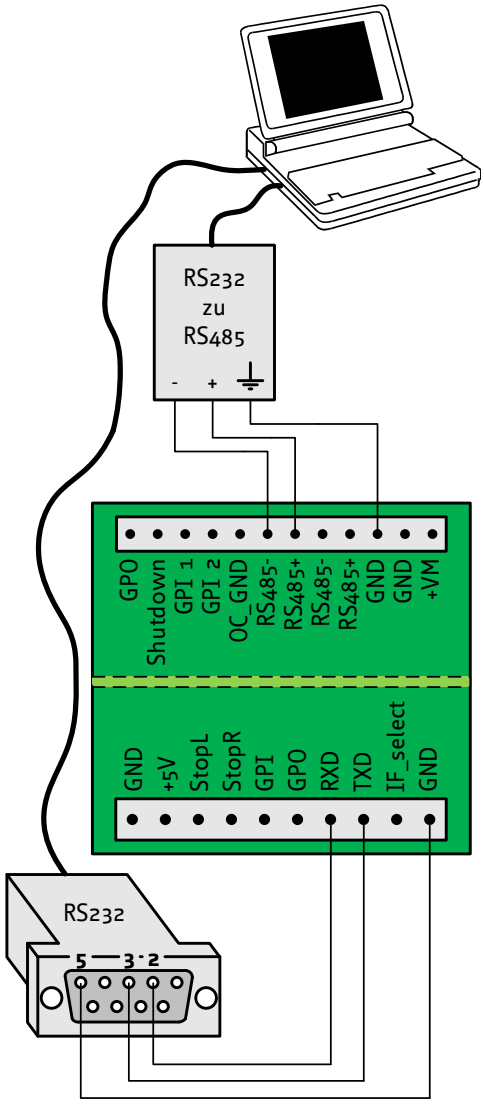
- PDx-109-57 V2
- Interface: Either use RS232 directly from PC or RS485 with a converter (add termination network and set telegram pause time if necessary).
- Nominal supply voltage +24V... 48V DC for your module
- TMCL-IDE program and PC

Precautions:

- Do not mix up connections or short-circuit pins.
- Avoid bounding I/O wires with motor power wires as this may cause noise picked up from the motor supply.
- Do not exceed the maximum power supply of 55V DC.
- ***Do not connect or disconnect the motor while powered!***
- ***Start with power supply OFF!***

## 5.1 Starting up

The following figure will show you which connectors have to be used.




1. **Connect the interface RS232 or RS485**

   a. RS232
   Use connector 2 for connecting the RS232 interface. Please have a look to the figure above how to do it.

   b. RS485
   Use connector 1 for connecting the RS485 interface. Using the RS485 interface has to be enabled via the interface selection input (connector2, terminal 2: IF select).

   Interface selection (IF): - leave open for RS232
   - connect to ground for RS485

2. **Connect the power supply**

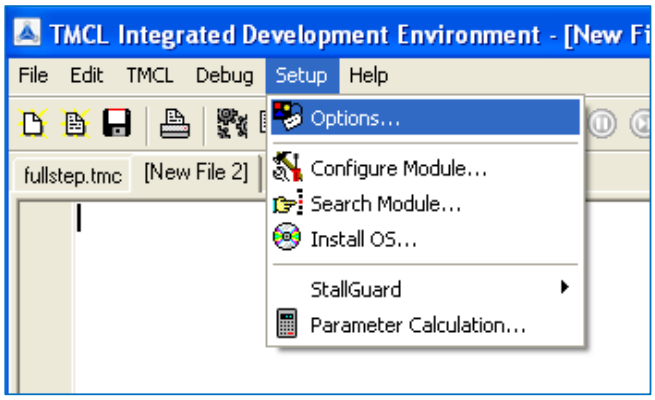
   Use connector 1 for connecting the power supply to PDx-109-57 VS.

3. **Switch ON the power supply**

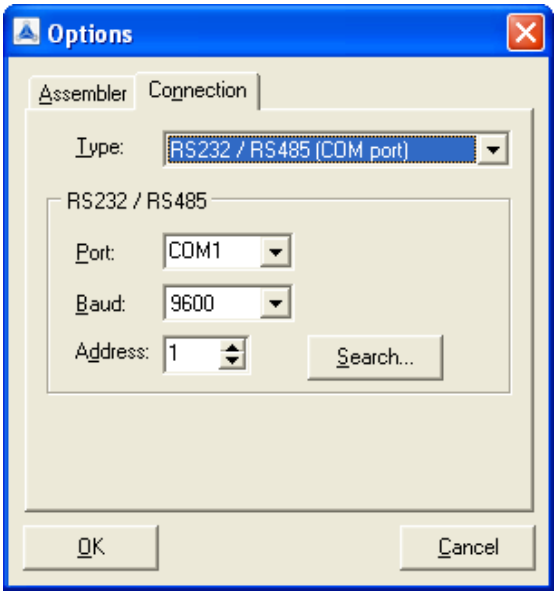
   The power LED flashes now.
   ***If this does not occur, switch power OFF and check your connections as well as the power supply.***

4. **Start the TMCL-IDE software development environment**
   The TMCL-IDE is available on the TechLibCD and on www.trinamic.com.

   Installing the TMCL-IDE:
   - Make sure the COM port you intend to use is not blocked by another program.
   - Open TMCL-IDE by clicking ***TMCL.exe***.
   - Choose ***Setup*** and ***Options*** and thereafter the ***Connection tab***.



- Choose ***COM port*** and ***type*** with fitting parameters (baud rate 9600 for RS232). Click ***OK***.

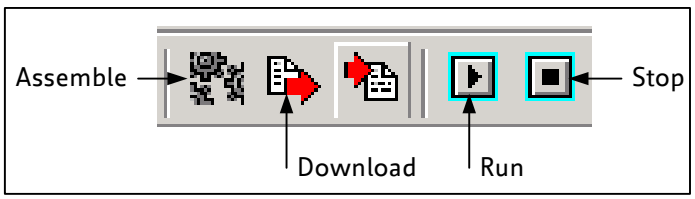



## 5.2 Testing with a simple TMCL™ program

Open the file test2.tmc. The following source code appears on the screen:

A description for the TMCL™ commands can be found in Appendix A.

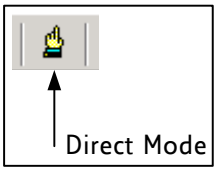
```
//A simple example for using TMCL™ and TMCL-IDE

        ROL 0, 500                  //Rotate motor 0 with speed 500
        WAIT TICKS, 0, 500
        MST 0
        ROR 0, 250                  //Rotate motor 1 with 250
        WAIT TICKS, 0, 500
        MST 0

        SAP 4, 0, 500               //Set max. Velocity
        SAP 5, 0, 50                //Set max. Acceleration
Loop:   MVP ABS, 0, 10000           //Move to Position 10000
        WAIT POS, 0, 0              //Wait until position reached
        MVP ABS, 0, -10000          //Move to Position -10000
        WAIT POS, 0, 0              //Wait until position reached
        JA Loop                     //Infinite Loop
```




1. Click on icon ***Assemble*** to convert the TMCL™ into binary code.
2. Then download the program to the TMCM-109 module via the icon ***Download***.
3. Press icon ***Run***. The desired program will be executed.
4. Click ***Stop*** button to stop the program.

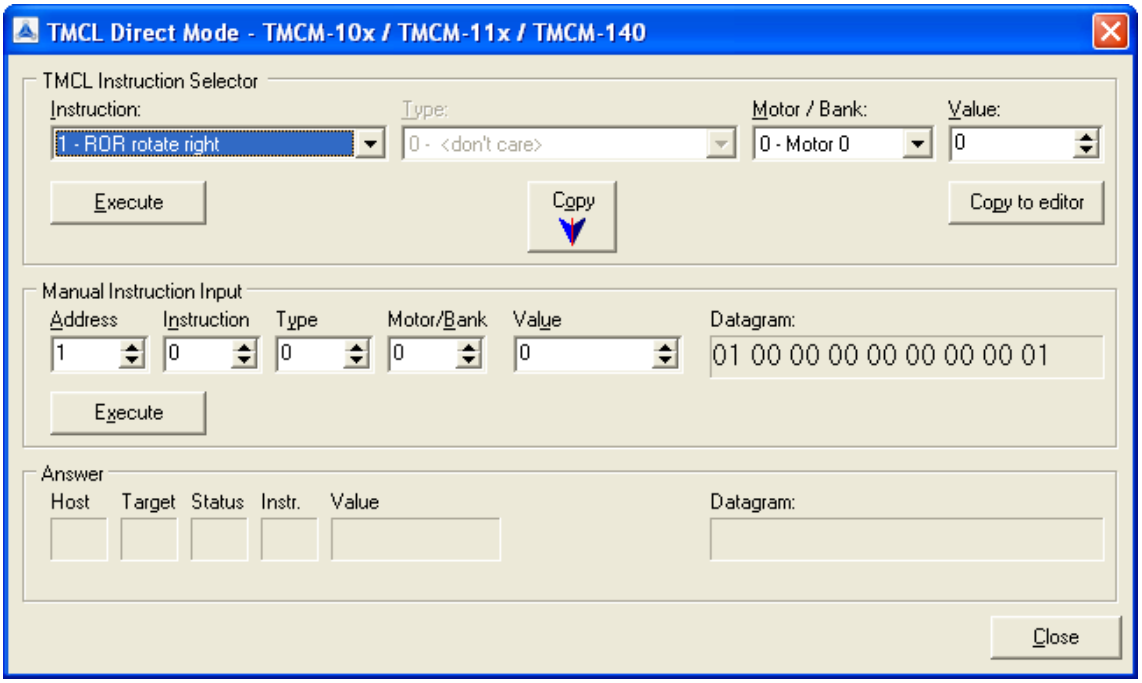
## 5.3 Operating the PANdrive™ in direct mode

1. Start TMCL™ ***Direct Mode***.




2. If the communication is established the TMCM-109 is automatically detected. ***If the module is not detected, please check all points above (cables, interface, power supply, COM port, baud rate).***

3. Issue a command by choosing ***instruction***, ***type*** (if necessary), ***motor***, and ***value*** and click ***Execute*** to send it to the module.




Examples:

- ROR rotate right, motor 0, value 500 -> Click *Execute*. The motor is rotating now.
- MST motor stop, motor 0 -> Click *Execute*. The motor stops now.

***Please use the TMCL-IDE axis parameter calculation tool for getting best values.***

# 6 TMCL™ and TMCL-IDE

The TMCM-109 module 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 the TMCL™ communication over RS232 and RS485 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 module. The TMCL™ interpreter on it 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 RS232/RS485 to the bus master. The master should not transfer the next command till then. 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-109 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.
- The mnemonic format is used for easy usage of the commands when developing stand-alone TMCL™ applications with the TMCL-IDE (IDE means *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.

## 6.1 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 or RS485 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.

**The binary command format for RS232 and RS485 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.

**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
```

## 6.2 Reply format

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

**The reply format for RS232 and RS485 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.

***Do not send the next command before you have received the reply!***

### 6.2.1 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 |

## 6.3 Stand-alone applications

The module is equipped with an EEPROM for storing TMCL™ applications. You can use the TMCL-IDE for developing stand-alone TMCL™ applications. You can load your program down into the EEPROM and then they 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.

## 6.4 TMCL™ command overview

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

### 6.4.1 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 |

### 6.4.2 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 |

### 6.4.3 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 |

### 6.4.4 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 |

### 6.4.5 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.

## 6.5 TMCL™ List of commands

The following TMCL™ commands are currently supported:

| Command | Number | Parameter | Description |
|---|---|---|---|
| ROR | 1 | <motor number>, <velocity> | Rotate right with specified velocity |
| ROL | 2 | <motor number>, <velocity> | Rotate left with specified velocity |
| MST | 3 | <motor number> | Stop motor movement |
| MVP | 4 | ABS\|REL\|COORD, <motor number>, <position\|offset> | Move to position (absolute or relative) |
| SAP | 5 | <parameter>, <motor number>, <value> | Set axis parameter (motion control specific settings) |
| GAP | 6 | <parameter>, <motor number> | Get axis parameter (read out motion control specific settings) |
| STAP | 7 | <parameter>, <motor number> | Store axis parameter permanently (non volatile) |
| RSAP | 8 | <parameter>; <motor number> | Restore axis parameter |
| SGP | 9 | <parameter>, <bank number>, <value> | Set global parameter (module specific settings, e.g. communication settings, or TMCL™ user variables) |
| GGP | 10 | <parameter>, <bank number> | Get global parameter (read out module specific settings e.g. communication settings, or TMCL™ user variables) |
| STGP | 11 | <parameter>, <bank number> | Store global parameter (TMCL™ user variables only) |
| RSGP | 12 | <parameter>, <bank> | Restore global parameter (TMCL™ user variables only) |
| RFS | 13 | START\|STOP\|STATUS, <motor number> | Reference search |
| SIO | 14 | <port number>, <bank number>, <value> | Set digital output to specified value |
| GIO | 15 | <port number>, <bank number> | Get value of analogue/digital input |
| CALC | 19 | <operation>, <value> | Process accumulator & value |
| COMP | 20 | <value> | Compare accumulator <-> value |
| JC | 21 | <condition>, <jump address> | Jump conditional |
| JA | 22 | <jump address> | Jump absolute |
| CSUB | 23 | <subroutine address> | Call subroutine |
| RSUB | 24 | | Return from subroutine |
| WAIT | 27 | <condition>, <motor number>, <ticks> | Wait with further program execution |
| STOP | 28 | | Stop program execution |
| SCO | 30 | <coordinate number>, <motor number>, <position> | Set coordinate |

| Command | Number | Parameter | Description |
|---|---|---|---|
| GCO | 31 | <coordinate number>, <motor number> | Get coordinate |
| CCO | 32 | <coordinate number>, <motor number> | Capture coordinate |
| CALCX | 33 | <operation> | Process accumulator & X-register |
| AAP | 34 | <parameter>, <motor number> | Accumulator to axis parameter |
| AGP | 35 | <parameter>, <bank> | Accumulator to global parameter |

**TMCL™ control commands:**

| 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<br>1 - run from specified address | (don't care) | (don't care)<br><br>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:<br>0 – stop<br>1 – run<br>2 – step<br>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<br>1 – binary | (don't care) | (don't care) |
| 137 – restore factory settings | reset all settings stored in the EEPROM to their factory defaults<br>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 6.6) | (don't care) | (don't care) | (don't care) |

## 6.6 The ASCII interface

TMCL™ also offers 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.***

### 6.6.1 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, 0, 50000
A MVP ABS, 0, 50000
AROL 2, 500
A MST 0
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.

### 6.6.2 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*.

### 6.6.3 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, UF0, UF1, UF2, UF3, UF4, UF5, UF6, and 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.

### 6.6.4 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 8.1). 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. 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.

## 6.7 Commands

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

### 6.7.1 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 TMC428-I stepper motor controller and the TMC249 power driver. This makes possible choosing a velocity between 0 and 2047.

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

**Mnemonic:** ROR <motor number>, <velocity>

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 1 | (don't care) | 0* | <velocity><br>0... 2047 |

*** 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, 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 Byte3 | Operand Byte2 | Operand Byte1 | Operand Byte0 | Checksum |
| **Value (hex)** | $01 | $01 | $00 | $00 | $00 | $00 | $01 | $5e | $61 |

## 6.7.2 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 TMC428-I stepper motor controller and the TMC249 power driver. This makes possible choosing a velocity between 0 and 2047.

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

**Mnemonic:** ROL <motor number>, <velocity>

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 2 | (don't care) | 0* | <velocity><br>0... 2047 |

*** 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, velocity = 1200
*Mnemonic:* ROL 0, 1200

*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 | $01 | $00 | $00 | $00 | $00 | $04 | $b0 | $b6 |

### 6.7.3 MST (motor stop)

With this command the motor will be instructed to stop with deceleration ramp (soft stop). For information about hard stops refer to chapter 9 (hints and tips) please.

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

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

**Mnemonic:** MST <motor number>

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 3 | (don't care) | 0* | (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 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 | $03 | $00 | $00 | $00 | $00 | $00 | $00 | $04 |

### 6.7.4 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 - 8388608 to +8388607 ($-2^{23}$ to$+2^{23}-1$).
- 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).

***Please note, that the distance between the actual position and the new one should not be more than 8388607 microsteps. Otherwise the motor will run in the wrong direction for taking a shorter way. If the value is exactly 8388608 the motor maybe stops.***

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

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

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

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 4 | 0 ABS – absolute | 0* | <position> |
| | 1 REL – relative | 0* | <offset> |
| | 2 COORD – coordinate | 0* | <coordinate number (0…20) |

*** Motor number is always 0 as only one motor is involved.**

**Reply in direct mode:**

| STATUS | VALUE |
|---|---|
| 100 – OK | (don't care) |

**Example MVP ABS:**

Move motor to (absolute) position 9000
*Mnemonic:* MVP ABS, 0, 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 | $00 | $23 | $28 | $50 |

**Example MVP REL:**

Move motor from current position 1000 steps backward (move relative -1000)
*Mnemonic:* MVP REL, 0, -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 |

**Examples MVP COORD:**

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 | $0f |

***It is possible to use stall detection. Please see section 9.2 for details.***

***When moving to a coordinate, the coordinate has to be set properly in advance with the help of the SCO command (see 6.7.24).***

## 6.7.5 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>, <motor number>, <value>

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 5 | <parameter number> | 0* | <value> |

*** Motor number is always 0 as only one motor is involved.**

**Reply in direct mode:**

| STATUS | VALUE |
|---|---|
| 100 – OK | (don't care) |

**List of parameters, which can be used for SAP:**

***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 [Unit] |
|---|---|---|---|
| 0 | target (next) position | The desired position in position mode (see ramp mode, no. 138). | $\pm 2^{23}$ [µsteps] |
| 1 | actual position | The current position of the motor. Should only be overwritten for reference point setting. | $\pm 2^{23}$ [µsteps] |
| 2 | target (next) speed | The desired speed in velocity mode (see ramp mode, no. 138). In position mode, this parameter is set by hardware: to the maximum speed during acceleration, and to zero during deceleration and rest. | ±2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 3 | actual speed | The current rotation speed. | ±2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 4 | maximum positioning speed | Should not exceed the physically highest possible value. Adjust the pulse divisor (no. 154), if the speed value is very low (<50) or above the upper limit. See TMC 428 datasheet for calculation of physical units. | 0...2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 5 | maximum acceleration | The limit for acceleration (and deceleration). Changing this parameter requires re-calculation of the acceleration factor (no. 146) and the acceleration divisor (no. 137), which is done automatically. See TMC 428 datasheet for calculation of physical units. | 0... 2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 6 | absolute max. current | The most important motor setting, since too high values might cause motor damage! The maximum value is 255 (which mean 100% of the maximum current of the module). | 0... 1500 [mA] |

| Number | Axis Parameter | Description | Range [Unit] |
|---|---|---|---|
| 7 | standby current | The current limit two seconds after the motor has stopped. | 0… 1500 [mA] |
| 12 | right limit switch disable | If set, deactivates the stop function of the right switch | 0/1 |
| 13 | left limit switch disable | Deactivates the stop function of the left switch resp. reference switch if set. | 0/1 |
| 130 | minimum speed | Should always be set 1 to ensure exact reaching of the target position. Normally no need to change! | 0…2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 138 | ramp mode | Automatically set when using ROR, ROL, MST and MVP.<br>0: position mode. Steps are generated, when the parameters actual position and target position differ. Trapezoidal speed ramps are provided.<br>2: velocity mode. The motor will run continuously and the speed will be changed with constant (maximum) acceleration, if the parameter *target speed* is changed.<br>For special purposes, the soft mode (value 1) with exponential decrease of speed can be selected. | 0/1/2 |
| 140 | microstep resolution | 0 – full step*)<br>1 – half step*)<br>2 – 4 microsteps<br>3 – 8 microsteps<br>4 – 16 microsteps<br>5 – 32 microsteps**)<br>6 – 64 microsteps**)<br>Note that modifying this parameter will affect the rotation speed in the same relation:<br>*) The full-step setting and the half-step setting are not optimized for use without an adapted microstepping table. These settings just step through the microstep table in steps of 64 respectively 32. To get real full stepping use axis parameter 211 or load an adapted microstepping table.<br>**) If the module is specified for 16 microsteps only, switching to 32 or 64 microsteps brings an enhancement in resolution and smoothness. The position counter will use the full resolution, but, however, the motor will resolve a maximum of 24 different microsteps only for the 32 or 64 microstep units. | 0… 6 |
| 141 | ref. switch tolerance | For three-switch mode: a position range, where an additional switch (connected to the REFL input) won't cause motor stop. See section 9.1 for details. | 0… 4095 |
| 149 | soft stop flag | If cleared, the motor will stop immediately (disregarding motor limits), when the reference or limit switch is hit. | 0/1 |
| 153 | ramp divisor | The exponent of the scaling factor for the ramp generator- should be de/incremented carefully (in steps of one). | 0… 13 |

| Number | Axis Parameter | Description | Range [Unit] |
|---|---|---|---|
| 154 | pulse divisor | The exponent of the scaling factor for the pulse (step) generator – should be de/incremented carefully (in steps of one). | 0… 13 |
| 193 | referencing mode | 1 – Only the left reference switch is searched.<br>2 – The right switch is searched and afterwards the left switch is searched.<br>3 – Three-switch-mode: the right switch is searched first and afterwards the reference switch will be searched.<br>Please see chapter 6.7.13 for details on reference search. | 1/2/3 |
| 194 | referencing search speed | For the reference search this value specifies the search speed as a fraction of the maximum velocity:<br>0 – full speed<br>1 – half of the maximum speed<br>2 – a quarter of the maximum speed<br>3 – 1/8 of the maximum speed (etc.) | 0… 8 |
| 195 | referencing switch speed | Similar to parameter no. 194, the speed for the switching point calibration can be selected. | 0… 8 |
| 203 | mixed decay threshold | If the actual velocity is above this threshold, mixed decay will be used. This can also be set to -1 which turns on mixed decay permanently also in the rising part of the microstep wave. This can be used to fix microstep errors. | 0… 2048<br>or -1<br>$\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 204 | freewheeling | Time after which the power to the motor will be cut when its velocity has reached zero. | 0…65535<br>0 = never<br>[msec] |
| 205 | stall detection threshold | Stall detection threshold. Set it to 0 for no stall detection or to a value between 1 (low threshold) and 7 (high threshold). The motor will be stopped if the load value exceeds the stall detection threshold. Switch off mixed decay to get usable results. | 0… 7 |
| 211 | fullstep threshold | When exceeding this speed the driver will switch to real full step mode. To disable this feature set this parameter to zero or to a value greater than 2047.<br>Setting a full step threshold allows higher motor torque of the motor at higher velocity. When experimenting with this in a given application, try to reduce the motor current in order to be able to reach a higher motor velocity! | 0…2048<br>$\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 214 | power down delay | Standstill period before the current is changed down to standby current. The standard value is 200 (value equates 2000msec). | 1… 65535<br>[10msec] |

***Please use the TMCL-IDE axis parameter calculation tool for getting best values.***

**Example:**

Set the absolute maximum current 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 | $d4 |

### 6.7.6 GAP (get axis parameter)

Most parameters of the TMCM-109 can be adjusted individually. 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>, <motor number>

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 6 | <parameter number> | <motor number>* | (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) |

**List of parameters, which can be used for GAP:**

| Number | Axis Parameter | Description | Range [Unit] |
|---|---|---|---|
| 0 | target (next) position | The desired position in position mode (see ramp mode, no. 138). | $\pm 2^{23}$ [µsteps] |
| 1 | actual position | The current position of the motor. Should only be overwritten for reference point setting. | $\pm 2^{23}$ [µsteps] |
| 2 | target (next) speed | The desired speed in velocity mode (see ramp mode, no. 138). In position mode, this parameter is set by hardware: to the maximum speed during acceleration, and to zero during deceleration and rest. | ±2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 3 | actual speed | The current rotation speed. | ±2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 4 | maximum positioning speed | Should not exceed the physically highest possible value. Adjust the pulse divisor (no. 154), if the speed value is very low (<50) or above the upper limit. See TMC 428 datasheet for calculation of physical units. | 0...2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 5 | maximum acceleration | The limit for acceleration (and deceleration). Changing this parameter requires re-calculation of the acceleration factor (no. 146) and the acceleration divisor (no. 137), which is done automatically. See TMC 428 datasheet for calculation of physical units. | 0... 2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 6 | absolute max. current | The most important motor setting, since too high values might cause motor damage! The maximum value is 255 (which mean 100% of the maximum current of the module). | 0... 1500 [mA] |
| 7 | standby current | The current limit two seconds after the motor has stopped. | 0... 1500 [mA] |

| Number | Axis Parameter | Description | Range [Unit] |
|---|---|---|---|
| 8 | target pos. reached | Indicates that the actual position equals the target position. | 0/1 |
| 9 | ref. switch status | The logical state of the reference (left) switch. See the TMC 428 data sheet for the different switch modes. The default has two switch modes: the left switch as the reference switch, the right switch as a limit (stop) switch. | 0/1 |
| 10 | right limit switch status | The logical state of the (right) limit switch. | 0/1 |
| 11 | left limit switch status | The logical state of the left limit switch (in three switch mode) | 0/1 |
| 12 | right limit switch disable | If set, deactivates the stop function of the right switch | 0/1 |
| 13 | left limit switch disable | Deactivates the stop function of the left switch resp. reference switch if set. | 0/1 |
| 130 | minimum speed | Should always be set 1 to ensure exact reaching of the target position. Normally no need to change! | 0…2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ |
| 135 | actual acceleration | The current acceleration (read only). | 0… 2047* |
| 138 | ramp mode | Automatically set when using ROR, ROL, MST and MVP.<br>0: position mode. Steps are generated, when the parameters actual position and target position differ. Trapezoidal speed ramps are provided.<br>2: velocity mode. The motor will run continuously and the speed will be changed with constant (maximum) acceleration, if the parameter *target speed* is changed.<br>For special purposes, the soft mode (value 1) with exponential decrease of speed can be selected. | 0/1/2 |

| Number | Axis Parameter | Description | Range [Unit] |
|---|---|---|---|
| 140 | microstep resolution | 0 – full step*)<br>1 – half step*)<br>2 – 4 microsteps<br>3 – 8 microsteps<br>4 – 16 microsteps<br>5 – 32 microsteps**)<br>6 – 64 microsteps**)<br>Note that modifying this parameter will affect the rotation speed in the same relation:<br>*) The full-step setting and the half-step setting are not optimized for use without an adapted microstepping table. These settings just step through the microstep table in steps of 64 respectively 32. To get real full stepping use axis parameter 211 or load an adapted microstepping table.<br>**) If the module is specified for 16 microsteps only, switching to 32 or 64 microsteps brings an enhancement in resolution and smoothness. The position counter will use the full resolution, but, however, the motor will resolve a maximum of 24 different microsteps only for the 32 or 64 microstep units. | 0…6 |
| 141 | ref. switch tolerance | For three-switch mode: a position range, where an additional switch (connected to the REFL input) won't cause motor stop. See section 9.1 for details. | 0…4095 |
| 149 | soft stop flag | If cleared, the motor will stop immediately (disregarding motor limits), when the reference or limit switch is hit. | 0/1 |
| 153 | ramp divisor | The exponent of the scaling factor for the ramp generator- should be de/incremented carefully (in steps of one). | 0…13 |
| 154 | pulse divisor | The exponent of the scaling factor for the pulse (step) generator – should be de/incremented carefully (in steps of one). | 0…13 |
| 193 | referencing mode | 1 – Only the left reference switch is searched.<br>2 – The right switch is searched and afterwards the left switch is searched.<br>3 – Three-switch-mode: the right switch is searched first and afterwards the reference switch will be searched.<br>Please see chapter 6.7.13 for details on reference search. | 1/2/3 |
| 194 | referencing search speed | For the reference search this value specifies the search speed as a fraction of the maximum velocity:<br>0 – full speed<br>1 – half of the maximum speed<br>2 – a quarter of the maximum speed<br>3 – 1/8 of the maximum speed (etc.) | 0…8 |
| 195 | referencing switch speed | Similar to parameter no. 194, the speed for the switching point calibration can be selected. | 0..8 |

<table>
<tr><th>Number</th><th>Axis Parameter</th><th>Description</th><th>Range [Unit]</th></tr>
<tr><td>203</td><td>mixed decay threshold</td><td>If the actual velocity is above this threshold, mixed decay will be used. This can also be set to -1 which turns on mixed decay permanently also in the rising part of the microstep wave. This can be used to fix microstep errors.</td><td>0..2048<br>or -1<br>$\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$</td></tr>
<tr><td>204</td><td>freewheeling</td><td>Time after which the power to the motor will be cut when its velocity has reached zero.</td><td>0…65535<br>0 = never<br>[msec]</td></tr>
<tr><td>205</td><td>stall detection threshold</td><td>Stall detection threshold. Set it to 0 for no stall detection or to a value between 1 (low threshold) and 7 (high threshold). The motor will be stopped if the load value exceeds the stall detection threshold. Switch off mixed decay to get usable results.</td><td>0…7</td></tr>
<tr><td>206</td><td>actual load value</td><td>Readout of the actual load value used for stall detection.</td><td>0…7</td></tr>
<tr><td>208</td><td>Driver Error Flags of TMC249</td><td>
<table>
<tr><th>Bit</th><th>Name</th><th>Function</th><th>Remark</th></tr>
<tr><td>7</td><td>OT</td><td>Overtemperature</td><td>1 = chip of due to overtemperature</td></tr>
<tr><td>6</td><td>OTPW</td><td>Temperature prewarning</td><td>1= prewarning temperature exceeded</td></tr>
<tr><td>5</td><td>UV</td><td>Driver undervoltage</td><td>1 = undervoltage on VS</td></tr>
<tr><td>4</td><td>OCHS</td><td>Overcurrent high side</td><td>3 PWM cycles with overcurrent within 63 PWM cycles</td></tr>
<tr><td>3</td><td>OLB</td><td>Open load bridge B</td><td>No PWM switch off for 14 oscillator cycles</td></tr>
<tr><td>2</td><td>OLA</td><td>Open load bridge A</td><td>No PWM switch off for 14 oscillator cycles</td></tr>
<tr><td>1</td><td>OCB</td><td>Overcurrent bridge B low side</td><td>3 PWM cycles with overcurrent within 63 PWM cycles</td></tr>
<tr><td>0</td><td>OCA</td><td>Overcurrent bridge A low side</td><td>3 PWM cycles with overcurrent within 63 PWM cycles</td></tr>
</table>
</td><td>0… 7</td></tr>
<tr><td>211</td><td>fullstep threshold</td><td>When exceeding this speed the driver will switch to real full step mode. To disable this feature set this parameter to zero or to a value greater than 2047.<br>Setting a full step threshold allows higher motor torque of the motor at higher velocity. When experimenting with this in a given application, try to reduce the motor current in order to be able to reach a higher motor velocity!</td><td>0..2048<br>$\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$</td></tr>
<tr><td>214</td><td>power down delay</td><td>Standstill period before the current is changed down to standby current. The standard value is 200 (value equates 2000msec).</td><td>1… 65535<br>[10msec]</td></tr>
</table>

* Unit of acceleration: $\frac{16MHz^2}{536870912 \cdot 2^{puls\_divisor+ramp\_divisor}} \frac{\text{microsteps}}{\text{sec}^2}$

**Example:**

Get the actual position of motor
*Mnemonic:* GAP 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 | $06 | $01 | $00 | $00 | $00 | $00 | $00 | $08 |

*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**

### 6.7.7 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 (refer to axis parameter list in chapter 7).

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

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

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

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 7 | <parameter number> | <motor number>*[1] | (don't care)*[2] |

***[1] Motor number is always 0 as only one motor is involved.**
***[2] 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 7 | 0 | s. chapter 7 |

**List of parameters, which can be used for STAP:**

| Number | Axis Parameter | Description |
|---|---|---|
| 4 | maximum positioning speed | Should not exceed the physically highest possible value. Adjust the pulse divisor (no. 154), if the speed value is very low (<50) or above the upper limit. See TMC 428 datasheet for calculation of physical units. |
| 5 | maximum acceleration | The limit for acceleration (and deceleration). Changing this parameter requires re-calculation of the acceleration factor (no. 146) and the acceleration divisor (no. 137), which is done automatically. See TMC 428 datasheet for calculation of physical units. |
| 6 | absolute max. current | The most important motor setting, since too high values might cause motor damage! The maximum value is 255 (which mean 100% of the maximum current of the module). |
| 7 | standby current | The current limit two seconds after the motor has stopped. |
| 12 | right limit switch disable | If set, deactivates the stop function of the right switch |
| 13 | left limit switch disable | Deactivates the stop function of the left switch resp. reference switch if set. |
| 130 | minimum speed | Should always be set 1 to ensure exact reaching of the target position. Normally no need to change! |

| Number | Axis Parameter | Description |
|---|---|---|
| 138 | ramp mode | Automatically set when using ROR, ROL, MST and MVP.<br>0: position mode. Steps are generated, when the parameters actual position and target position differ. Trapezoidal speed ramps are provided.<br>2: velocity mode. The motor will run continuously and the speed will be changed with constant (maximum) acceleration, if the parameter *target speed* is changed.<br>For special purposes, the soft mode (value 1) with exponential decrease of speed can be selected. |
| 140 | microstep resolution | 0 – full step*)<br>1 – half step*)<br>2 – 4 microsteps<br>3 – 8 microsteps<br>4 – 16 microsteps<br>5 – 32 microsteps**)<br>6 – 64 microsteps**)<br>Note that modifying this parameter will affect the rotation speed in the same relation:<br>*) The full-step setting and the half-step setting are not optimized for use without an adapted microstepping table. These settings just step through the microstep table in steps of 64 respectively 32. To get real full stepping use axis parameter 211 or load an adapted microstepping table.<br>**) If the module is specified for 16 microsteps only, switching to 32 or 64 microsteps brings an enhancement in resolution and smoothness. The position counter will use the full resolution, but, however, the motor will resolve a maximum of 24 different microsteps only for the 32 or 64 microstep units. |
| 149 | soft stop flag | If cleared, the motor will stop immediately (disregarding motor limits), when the reference or limit switch is hit. |
| 153 | ramp divisor | The exponent of the scaling factor for the ramp generator- should be de/incremented carefully (in steps of one). |
| 154 | pulse divisor | The exponent of the scaling factor for the pulse (step) generator – should be de/incremented carefully (in steps of one). |
| 193 | referencing mode | 1 – Only the left reference switch is searched.<br>2 – The right switch is searched and afterwards the left switch is searched.<br>3 – Three-switch-mode: the right switch is searched first and afterwards the reference switch will be searched.<br>Please see chapter 6.7.13 for details on reference search. |

| Number | Axis Parameter | Description |
|---|---|---|
| 194 | referencing search speed | For the reference search this value specifies the search speed as a fraction of the maximum velocity:<br>0 – full speed<br>1 – half of the maximum speed<br>2 – a quarter of the maximum speed<br>3 – 1/8 of the maximum speed (etc.) |
| 195 | referencing switch speed | Similar to parameter no. 194, the speed for the switching point calibration can be selected. |
| 203 | mixed decay threshold | If the actual velocity is above this threshold, mixed decay will be used. This can also be set to -1 which turns on mixed decay permanently also in the rising part of the microstep wave. This can be used to fix microstep errors. |
| 204 | freewheeling | Time after which the power to the motor will be cut when its velocity has reached zero. |
| 205 | stall detection threshold | Stall detection threshold. Set it to 0 for no stall detection or to a value between 1 (low threshold) and 7 (high threshold). The motor will be stopped if the load value exceeds the stall detection threshold. Switch off mixed decay to get usable results. |
| 211 | fullstep threshold | When exceeding this speed the driver will switch to real full step mode. To disable this feature set this parameter to zero or to a value greater than 2047.<br>Setting a full step threshold allows higher motor torque of the motor at higher velocity. When experimenting with this in a given application, try to reduce the motor current in order to be able to reach a higher motor velocity! |
| 214 | power down delay | Standstill period before the current is changed down to standby current. The standard value is 200 (value equates 2000msec). |

**Example:**

Store the maximum speed
*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 | $0C |

***Note: The STAP command will not have any effect when the configuration EEPROM is locked (refer to 8.1). In direct mode, the error code 5 (configuration EEPROM locked, see also section 6.2.1) will be returned in this case.***

### 6.7.8 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 (refer to axis parameter list in chapter 7). 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, and AAP

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

**Binary representation:**

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

*** Motor number is always 0 as only one motor is involved.**

**Reply structure in direct mode:**

| STATUS | VALUE |
|---|---|
| 100 – OK | (don't care) |

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

| Number | Axis Parameter | Description |
|---|---|---|
| 4 | maximum positioning speed | Should not exceed the physically highest possible value. Adjust the pulse divisor (no. 154), if the speed value is very low (<50) or above the upper limit. See TMC 428 datasheet for calculation of physical units. |
| 5 | maximum acceleration | The limit for acceleration (and deceleration). Changing this parameter requires re-calculation of the acceleration factor (no. 146) and the acceleration divisor (no. 137), which is done automatically. See TMC 428 datasheet for calculation of physical units. |
| 6 | absolute max. current | The most important motor setting, since too high values might cause motor damage! The maximum value is 255 (which mean 100% of the maximum current of the module). |
| 7 | standby current | The current limit two seconds after the motor has stopped. |
| 12 | right limit switch disable | If set, deactivates the stop function of the right switch |
| 13 | left limit switch disable | Deactivates the stop function of the left switch resp. reference switch if set. |
| 130 | minimum speed | Should always be set 1 to ensure exact reaching of the target position. Normally no need to change! |

| Number | Axis Parameter | Description |
|---|---|---|
| 138 | ramp mode | Automatically set when using ROR, ROL, MST and MVP.<br>0: position mode. Steps are generated, when the parameters actual position and target position differ. Trapezoidal speed ramps are provided.<br>2: velocity mode. The motor will run continuously and the speed will be changed with constant (maximum) acceleration, if the parameter *target speed* is changed.<br>For special purposes, the soft mode (value 1) with exponential decrease of speed can be selected. |
| 140 | microstep resolution | 0 – full step*)<br>1 – half step*)<br>2 – 4 microsteps<br>3 – 8 microsteps<br>4 – 16 microsteps<br>5 – 32 microsteps**)<br>6 – 64 microsteps**)<br>Note that modifying this parameter will affect the rotation speed in the same relation:<br>*) The full-step setting and the half-step setting are not optimized for use without an adapted microstepping table. These settings just step through the microstep table in steps of 64 respectively 32. To get real full stepping use axis parameter 211 or load an adapted microstepping table.<br>**) If the module is specified for 16 microsteps only, switching to 32 or 64 microsteps brings an enhancement in resolution and smoothness. The position counter will use the full resolution, but, however, the motor will resolve a maximum of 24 different microsteps only for the 32 or 64 microstep units. |
| 149 | soft stop flag | If cleared, the motor will stop immediately (disregarding motor limits), when the reference or limit switch is hit. |
| 153 | ramp divisor | The exponent of the scaling factor for the ramp generator- should be de/incremented carefully (in steps of one). |
| 154 | pulse divisor | The exponent of the scaling factor for the pulse (step) generator – should be de/incremented carefully (in steps of one). |
| 193 | referencing mode | 1 – Only the left reference switch is searched.<br>2 – The right switch is searched and afterwards the left switch is searched.<br>3 – Three-switch-mode: the right switch is searched first and afterwards the reference switch will be searched.<br>Please see chapter 6.7.13 for details on reference search. |

| Number | Axis Parameter | Description |
|---|---|---|
| 194 | referencing search speed | For the reference search this value specifies the search speed as a fraction of the maximum velocity:<br>0 – full speed<br>1 – half of the maximum speed<br>2 – a quarter of the maximum speed<br>3 – 1/8 of the maximum speed (etc.) |
| 195 | referencing switch speed | Similar to parameter no. 194, the speed for the switching point calibration can be selected. |
| 203 | mixed decay threshold | If the actual velocity is above this threshold, mixed decay will be used. This can also be set to -1 which turns on mixed decay permanently also in the rising part of the microstep wave. This can be used to fix microstep errors. |
| 204 | freewheeling | Time after which the power to the motor will be cut when its velocity has reached zero. |
| 205 | stall detection threshold | Stall detection threshold. Set it to 0 for no stall detection or to a value between 1 (low threshold) and 7 (high threshold). The motor will be stopped if the load value exceeds the stall detection threshold. Switch off mixed decay to get usable results. |
| 211 | fullstep threshold | When exceeding this speed the driver will switch to real full step mode. To disable this feature set this parameter to zero or to a value greater than 2047.<br>Setting a full step threshold allows higher motor torque of the motor at higher velocity. When experimenting with this in a given application, try to reduce the motor current in order to be able to reach a higher motor velocity! |
| 214 | power down delay | Standstill period before the current is changed down to standby current. The standard value is 200 (value equates 2000msec). |

**Example:**

Restore the maximum current
*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 | $0f |

### 6.7.9 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, 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:**

| Number | Global parameter | Description | | | Range |
|---|---|---|---|---|---|
| 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 |
| 65 | RS232 baud rate | 0 | 9600 baud | *Default* | 0…7 |
| | | 1 | 14400 baud | | |
| | | 2 | 19200 baud | | |
| | | 3 | 28800 baud | | |
| | | 4 | 38400 baud | | |
| | | 5 | 57600 baud | | |
| | | 6 | 76800 baud | *Not supported by Windows!* | |
| | | 7 | (115200 baud) | *3.68% Error (111111 Bits/s)* | |
| 66 | serial address | The module (target) address for RS-232. | | | 0…255 |
| 67 | ASCII mode | Configure the TMCL™ ASCII interface:<br>Bit 0: 0 – start up in binary (normal) mode<br>1 – start up in ASCII mode<br>Bits 4 and 5:<br>00 – Echo back each character<br>01 – Echo back complete command<br>10 – Do not send echo, only send command reply | | | |
| 73 | configuration EEPROM lock flag | Write: 1234 to lock the EEPROM, 4321 to unlock it.<br>Read: 1=EEPROM locked, 0=EEPROM unlocked. | | | 0/1 |
| 75 | telegram pause time | Pause time before the reply via RS232 is sent. For RS232 set to 0. | | | 0…255 |
| 76 | serial host address | Host address used in the reply telegrams sent back via RS232. | | | 0…255 |

| Number | Global parameter | Description | Range |
|---|---|---|---|
| 77 | auto start mode | 0: Do not start TMCL™ application after power up (default).<br>1: Start TMCL™ application automatically after power up. | 0/1 |
| 80 | shutdown pin functionality | Select the functionality of the SHUTDOWN pin<br>0 – no function<br>1 – high active<br>2 – low active | 0…2 |
| 81 | TMCL™ code protection | Protect a TMCL™ program against disassembling or overwriting.<br>0 – no protection<br>1 – protection against disassembling<br>2 – protection against overwriting<br>3 – protection against disassembling and overwriting<br>***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.*** | 0,1,2,3 |
| 132 | tick timer | A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value. | |

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

The global parameter bank 1 is normally not available. It may be used for customer specific extensions of the firmware. Together with user definable commands (see section 7.3) these variables form the interface between extensions of the firmware (written in C) and TMCL™ applications.

**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 |
|---|---|---|---|
| 0 | general purpose variable #0 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 1 | general purpose variable #1 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 2 | general purpose variable #2 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 3 | general purpose variable #3 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 4 | general purpose variable #4 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 5 | general purpose variable #5 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 6 | general purpose variable #6 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 7 | general purpose variable #7 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 8 | general purpose variable #8 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 9 | general purpose variable #9 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 10 | general purpose variable #10 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 11 | general purpose variable #11 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 12 | general purpose variable #12 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 13 | general purpose variable #13 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 14 | general purpose variable #14 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 15 | general purpose variable #15 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 16 | general purpose variable #16 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 17 | general purpose variable #17 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 18 | general purpose variable #18 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 19 | general purpose variable #19 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 20...55 | general purpose variables #20..#55 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |

**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 |

***Please refer to chapter 8 for more information about bank 0 to 2.***

## 6.7.10 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, bank 0 and 1 are used for global parameters, and bank 2 is used for user variables.

**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 | <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 GGP:**

| Number | Global parameter | Description | | | Range |
|---|---|---|---|---|---|
| 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 |
| 65 | RS232 baud rate | 0 | 9600 baud | *Default* | 0…7 |
| | | 1 | 14400 baud | | |
| | | 2 | 19200 baud | | |
| | | 3 | 28800 baud | | |
| | | 4 | 38400 baud | | |
| | | 5 | 57600 baud | | |
| | | 6 | 76800 baud | *Not supported by Windows!* | |
| | | 7 | (115200 baud) | *3.68% Error (111111 Bits/s)* | |
| 66 | serial address | The module (target) address for RS-232/RS-485. | | | 0…255 |
| 67 | ASCII mode | Configure the TMCL™ ASCII interface:<br>Bit 0: 0 – start up in binary (normal) mode<br>1 – start up in ASCII mode<br>Bits 4 and 5:<br>00 – Echo back each character<br>01 – Echo back complete command<br>10 – Do not send echo, only send command reply | | | |
| 73 | configuration EEPROM lock flag | Write: 1234 to lock the EEPROM, 4321 to unlock it.<br>Read: 1=EEPROM locked, 0=EEPROM unlocked. | | | 0/1 |
| 75 | telegram pause time | Pause time before the reply via RS232 is sent. For RS232 set to 0. | | | 0…255 |
| 76 | serial host address | Host address used in the reply telegrams sent back via RS232. | | | 0…255 |
| 77 | auto start mode | 0: Do not start TMCL™ application after power up (default).<br>1: Start TMCL™ application automatically after power up. | | | 0/1 |

| Number | Global parameter | Description | Range |
|---|---|---|---|
| 80 | shutdown pin functionality | Select the functionality of the SHUTDOWN pin<br>0 – no function<br>1 – high active<br>2 – low active | 0..2 |
| 81 | TMCL™ code protection | Protect a TMCL™ program against disassembling or overwriting.<br>0 – no protection<br>1 – protection against disassembling<br>2 – protection against overwriting<br>3 – protection against disassembling and overwriting<br>***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.*** | 0,1,2,3 |
| 128 | TMCL™ application status | 0 –stop<br>1 – run<br>2 – step<br>3 – reset | 0…3 |
| 129 | download mode | 0 – normal mode<br>1 – download mode | 0/1 |
| 130 | TMCL™ program counter | The index of the currently executed TMCL™ instruction. | |
| 132 | tick timer | A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value. | |
| 133 | random number | Choose a random number. ***Read only!*** | 0…2147483647 |

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

The global parameter bank 1 is normally not available. It may be used for customer specific extensions of the firmware. Together with user definable commands (see section 7.3) these variables form the interface between extensions of the firmware (written in C) and TMCL™ applications.

**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 |
|---|---|---|---|
| 0 | general purpose variable #0 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 1 | general purpose variable #1 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 2 | general purpose variable #2 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 3 | general purpose variable #3 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 4 | general purpose variable #4 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 5 | general purpose variable #5 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 6 | general purpose variable #6 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 7 | general purpose variable #7 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 8 | general purpose variable #8 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 9 | general purpose variable #9 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 10 | general purpose variable #10 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 11 | general purpose variable #11 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 12 | general purpose variable #12 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 13 | general purpose variable #13 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 14 | general purpose variable #14 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 15 | general purpose variable #15 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 16 | general purpose variable #16 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 17 | general purpose variable #17 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 18 | general purpose variable #18 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 19 | general purpose variable #19 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 20...55 | general purpose variables #20..#55 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |

**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**

***Please refer to chapter 8 for more information about bank 0 to 2.***

## 6.7.11 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.

**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 | <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 STGP:**

The global parameter bank 0 is not required for the STGP command, because these parameters are automatically stored with the SGP command in EEPROM.

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

The global parameter bank 1 is normally not available, but can be used in customer specific extensions of the firmware.

**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 |
|---|---|---|
| 0 | general purpose variable #0 | for use in TMCL™ applications |
| 1 | general purpose variable #1 | for use in TMCL™ applications |
| 2 | general purpose variable #2 | for use in TMCL™ applications |
| 3 | general purpose variable #3 | for use in TMCL™ applications |
| 4 | general purpose variable #4 | for use in TMCL™ applications |
| 5 | general purpose variable #5 | for use in TMCL™ applications |
| 6 | general purpose variable #6 | for use in TMCL™ applications |
| 7 | general purpose variable #7 | for use in TMCL™ applications |
| 8 | general purpose variable #8 | for use in TMCL™ applications |
| 9 | general purpose variable #9 | for use in TMCL™ applications |
| 10 | general purpose variable #10 | for use in TMCL™ applications |
| 11 | general purpose variable #11 | for use in TMCL™ applications |
| 12 | general purpose variable #12 | for use in TMCL™ applications |
| 13 | general purpose variable #13 | for use in TMCL™ applications |
| 14 | general purpose variable #14 | for use in TMCL™ applications |
| 15 | general purpose variable #15 | for use in TMCL™ applications |
| 16 | general purpose variable #16 | for use in TMCL™ applications |
| 17 | general purpose variable #17 | for use in TMCL™ applications |
| 18 | general purpose variable #18 | for use in TMCL™ applications |
| 19 | general purpose variable #19 | for use in TMCL™ applications |

| Number | Global parameter | Description |
|---|---|---|
| 20…55 | general purpose variables #20..#55 | for use in TMCL™ applications |

**Example:**

Store the user variable 5 in EEPROM.
STGP 5, 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 | $0b | $05 | $02 | $00 | $00 | $00 | $00 | $13 |

***Note: The STAP command will not have any effect when the configuration EEPROM is locked (refer to 8.1). In direct mode, the error code 5 (configuration EEPROM locked, see also section 6.2.1) will be returned in this case.***

***Please refer to chapter 8 for more information about bank 0 to 2.***

## 6.7.12 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 global parameter list in chapter 8). 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, and AAP

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

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 8 | <parameter number> | <bank number> | (don't care) |

**Reply structure in direct mode:**

| STATUS | VALUE |
|---|---|
| 100 – OK | (don't care) |

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

The global parameter bank 0 is not required for the STGP command, because these parameters are automatically stored with the SGP command in EEPROM.

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

The global parameter bank 1 is normally not available, but can be used in customer specific extensions of the firmware.

**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 |
|---|---|---|
| 0 | general purpose variable #0 | for use in TMCL™ applications |
| 1 | general purpose variable #1 | for use in TMCL™ applications |
| 2 | general purpose variable #2 | for use in TMCL™ applications |
| 3 | general purpose variable #3 | for use in TMCL™ applications |
| 4 | general purpose variable #4 | for use in TMCL™ applications |
| 5 | general purpose variable #5 | for use in TMCL™ applications |
| 6 | general purpose variable #6 | for use in TMCL™ applications |
| 7 | general purpose variable #7 | for use in TMCL™ applications |
| 8 | general purpose variable #8 | for use in TMCL™ applications |
| 9 | general purpose variable #9 | for use in TMCL™ applications |
| 10 | general purpose variable #10 | for use in TMCL™ applications |
| 11 | general purpose variable #11 | for use in TMCL™ applications |
| 12 | general purpose variable #12 | for use in TMCL™ applications |
| 13 | general purpose variable #13 | for use in TMCL™ applications |
| 14 | general purpose variable #14 | for use in TMCL™ applications |
| 15 | general purpose variable #15 | for use in TMCL™ applications |
| 16 | general purpose variable #16 | for use in TMCL™ applications |
| 17 | general purpose variable #17 | for use in TMCL™ applications |

| Number | Global parameter | Description |
|---|---|---|
| 18 | general purpose variable #18 | for use in TMCL™ applications |
| 19 | general purpose variable #19 | for use in TMCL™ applications |
| 20...55 | general purpose variables #20..#55 | for use in TMCL™ applications |

**Example:**

Restore the serial address of the device
*Mnemonic:* RSGP 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 | $0c | $42 | $00 | $00 | $00 | $00 | $00 | $4f |

***Please refer to chapter 8 for more information about bank 0 to 2.***

## 6.7.13 RFS (reference search)

The TMCM-109 module 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. 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>, <motor number>

**Binary representation:**

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

*** Motor number is always 0 as 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<br>other values – ref. search is active |

**Example:**

Start reference search

*Mnemonic:* RFS START, 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 | $0d | $00 | $00 | $00 | $00 | $00 | $00 | $0e |

***It is possible to use stall detection instead of a reference search. Please see section 9 for details.***

## 6.7.14 SIO (set output)

This command sets the status of the general digital output either to low (0) or to high (1).

**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:**

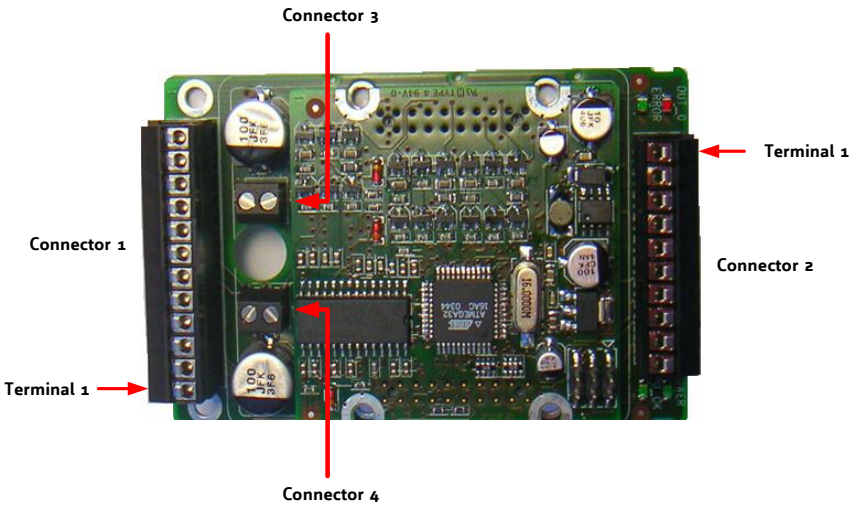
| STATUS | VALUE |
|---|---|
| 100 – OK | (don't care) |

**Example:**

Set OUT_1 to high (bank 2, output 1; general purpose output)
*Mnemonic:* SIO 0, 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 | $00 | $02 | $00 | $00 | $00 | $01 | $12 |




**Available I/O ports (connector 3):**

| Terminal | I/O port | Command | Range |
|---|---|---|---|
| 1 of connector 1 (same as terminal 5 of connector 2) | OUT_0 | SIO 0, <bank number>, 1/0 | 1/0 |

## 6.7.15 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 0
*Mnemonic:* GIO 0, 1

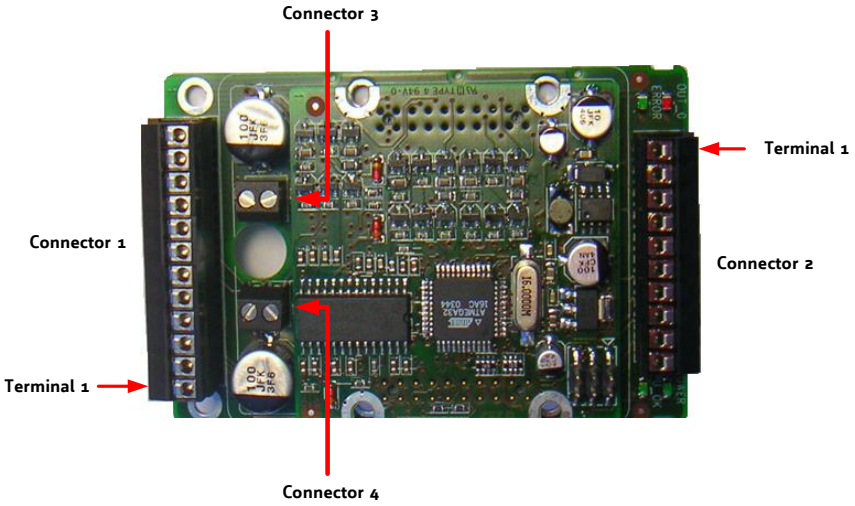
*Binary:*

| 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 | $0f | $00 | $01 | $00 | $00 | $00 | $00 | $11 |

*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




### 6.7.15.1 I/O bank 0 – digital inputs (connector 3):

***The ADIN lines can be read as digital or analogue inputs at the same time. The analogue values can be accessed in bank 1.***

| Terminal | I/O port | Command | Range |
|---|---|---|---|
| 6 of connector 2 | IN_0 | GIO 0, 0 | 1/0 |
| 3 of connector 1 | IN_1 | GIO 1, 0 | 1/0 |
| 4 of connector 1 | IN_2 | GIO 2, 0 | 1/0 |

### 6.7.15.2 I/O bank 1 – analogue inputs:

***The ADIN lines can be read back as digital or analogue inputs at the same time. The digital states can be accessed in bank 0.***

| Terminal | I/O port | Command | Range |
|---|---|---|---|
| 6 of connector 2 | IN_0 | GIO 0, 1 | 0… 1023 |

### 6.7.15.3 I/O bank 2 – the states of digital outputs

***The states of the OUT lines (that have been set by SIO commands) can be read back using bank 2.***

| Terminal | I/O port | Command | Range |
|---|---|---|---|
| 1 of connector 1 (same as terminal 5 of connector 2) | OUT_0 | GIO 0, 2, <n> | 1/0 |

## 6.7.16 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<br>1 SUB – subtract from accu<br>2 MUL – multiply accu by<br>3 DIV – divide accu by<br>4 MOD – modulo divide by<br>5 AND – logical and accu with<br>6 OR – logical or accu with<br>7 XOR – logical exor accu with<br>8 NOT – logical invert accu<br>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 |

## 6.7.17 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.

***The host address and the reply are only used to take the instruction to the TMCL™ program memory while the TMCL™ program loads down. 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 #2 is greater than or equal to 1000.

```
GAP 1, 2, 0     //get axis parameter, type: no. 1 (actual position), motor: 2, 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 |

## 6.7.18 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. This function is for stand-alone operation only.

***The host address and the reply are only used to take the instruction to the TMCL™ program memory while the TMCL™ program loads down. 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<br>1 NZ - not zero<br>2 EQ - equal<br>3 NE - not equal<br>4 GT - greater<br>5 GE - greater/equal<br>6 LT - lower<br>7 LE - lower/equal<br>8 ETO - time out error<br>9 EAL – external alarm<br>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 Byte3 | Operand Byte2 | Operand Byte1 | Operand Byte0 | Checksum |
| Value (hex) | $01 | $15 | $05 | $00 | $00 | $00 | $00 | $0a | $25 |

## 6.7.19 JA (jump always)

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

***The host address and the reply are only used to take the instruction to the TMCL™ program memory while the TMCL™ program loads down. 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 |

## 6.7.20 CSUB (call subroutine)

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

***The host address and the reply are only used to take the instruction to the TMCL™ program memory while the TMCL™ program loads down. 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 Byte3 | Operand Byte2 | Operand Byte1 | Operand Byte0 | Checksum |
| **Value (hex)** | $01 | $17 | $00 | $00 | $00 | $00 | $00 | $64 | $7C |

## 6.7.21 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.

***The host address and the reply are only used to take the instruction to the TMCL™ program memory while the TMCL™ program loads down. 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 have a look at the CSUB example below.

*Binary format of RSUB:*

| 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 | $18 | $00 | $00 | $00 | $00 | $00 | $00 | $19 |

## 6.7.22 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.

***The host address and the reply are only used to take the instruction to the TMCL™ program memory while the TMCL™ program loads down. 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>, <motor number>, <ticks>
where <condition> is TICKS|POS|REFSW|LIMSW|RFS

**Binary representation:**

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

***[1] One tick is 10msec (in standard firmware).**
***[2] Motor number is always 0 as only one motor is involved.**

**Example:**

Wait for motor 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 Byte3 | Operand Byte2 | Operand Byte1 | Operand Byte0 | Checksum |
| Value (hex) | $01 | $1b | $01 | $01 | $00 | $00 | $00 | $00 | $1d |

## 6.7.23 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.

***Every stand-alone TMCL™ program needs the STOP command at its end. 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 |

## 6.7.24 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 always stored in RAM only. 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>, <motor number>, <position>

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 30 | <coordinate number> (0... 20) | 0* | <position> ($-2^{23}$... $+2^{23}$) |

*** Motor number is always 0 as only one motor is involved.**

**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 Byte3 | Operand Byte2 | Operand Byte1 | Operand Byte0 | Checksum |
| **Value (hex)** | $01 | $1e | $01 | $00 | $00 | $00 | $03 | $e8 | $0b |

## 6.7.25 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 always 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>, <motor number>

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 31 | <coordinate number> (0... 20) | 0* | (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:**

Get value coordinate 1 of motor
*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 | $21 |

*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**

## 6.7.26 CCO (capture coordinate)

The actual position of the axis is copied to the selected coordinate variable.

***Please note that the coordinate number 0 is always stored in RAM only. 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>, <motor number>

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 32 | <coordinate number> (0…20) | 0* | (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:**

Store current position of the axis 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 Byte3 | Operand Byte2 | Operand Byte1 | Operand Byte0 | Checksum |
| **Value (hex)** | $01 | $20 | $03 | $00 | $00 | $00 | $00 | $00 | $24 |

## 6.7.27 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<br>1 SUB – subtract X register from accu<br>2 MUL – multiply accu by X register<br>3 DIV – divide accu by X-register<br>4 MOD – modulo divide accu by x-register<br>5 AND – logical and accu with X-register<br>6 OR – logical or accu with X-register<br>7 XOR – logical exor accu with X-register<br>8 NOT – logical invert X-register<br>9 LOAD – load accu to X-register<br>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 |

## 6.7.28 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.

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

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

**Binary representation:**

| INSTRUCTION NO. | TYPE | MOT/BANK | VALUE |
|---|---|---|---|
| 34 | <parameter number> | 0* | <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) |

**List of parameters, which can be used for AAP:**

| Number | Axis Parameter | Description |
|---|---|---|
| 0 | target (next) position | The desired position in position mode (see ramp mode, no. 138). |
| 1 | actual position | The current position of the motor. Should only be overwritten for reference point setting. |
| 2 | target (next) speed | The desired speed in velocity mode (see ramp mode, no. 138). In position mode, this parameter is set by hardware: to the maximum speed during acceleration, and to zero during deceleration and rest. |
| 3 | actual speed | The current rotation speed. |
| 4 | maximum positioning speed | Should not exceed the physically highest possible value. Adjust the pulse divisor (no. 154), if the speed value is very low (<50) or above the upper limit. See TMC 428 datasheet for calculation of physical units. |
| 5 | maximum acceleration | The limit for acceleration (and deceleration). Changing this parameter requires re-calculation of the acceleration factor (no. 146) and the acceleration divisor (no. 137), which is done automatically. See TMC 428 datasheet for calculation of physical units. |
| 6 | absolute max. current | The most important motor setting, since too high values might cause motor damage! The maximum value is 255 (which mean 100% of the maximum current of the module). |
| 7 | standby current | The current limit two seconds after the motor has stopped. |
| 12 | right limit switch disable | If set, deactivates the stop function of the right switch |
| 13 | left limit switch disable | Deactivates the stop function of the left switch resp. reference switch if set. |

| Number | Axis Parameter | Description |
|---|---|---|
| 130 | minimum speed | Should always be set 1 to ensure exact reaching of the target position. Do not change! |
| 138 | ramp mode | Automatically set when using ROR, ROL, MST and MVP.<br>0: position mode. Steps are generated, when the parameters actual position and target position differ. Trapezoidal speed ramps are provided.<br>2: velocity mode. The motor will run continuously and the speed will be changed with constant (maximum) acceleration, if the parameter *target speed* is changed.<br>For special purposes, the soft mode (value 1) with exponential decrease of speed can be selected. |
| 140 | microstep resolution | 0 – full step*)<br>1 – half step*)<br>2 – 4 microsteps<br>3 – 8 microsteps<br>4 – 16 microsteps<br>5 – 32 microsteps**)<br>6 – 64 microsteps**)<br>Note that modifying this parameter will affect the rotation speed in the same relation:<br>*) The full-step setting and the half-step setting are not optimized for use without an adapted microstepping table. These settings just step through the microstep table in steps of 64 respectively 32. To get real full stepping use axis parameter 211 or load an adapted microstepping table.<br>**) If the module is specified for 16 microsteps only, switching to 32 or 64 microsteps brings an enhancement in resolution and smoothness. The position counter will use the full resolution, but, however, the motor will resolve a maximum of 24 different microsteps only for the 32 or 64 microstep units. |
| 141 | ref. switch tolerance | For three-switch mode: a position range, where an additional switch (connected to the REFL input) won't cause motor stop. See section 9.1 for details. |
| 149 | soft stop flag | If cleared, the motor will stop immediately (disregarding motor limits), when the reference or limit switch is hit. |
| 153 | ramp divisor | The exponent of the scaling factor for the ramp generator- should be de/incremented carefully (in steps of one). |
| 154 | pulse divisor | The exponent of the scaling factor for the pulse (step) generator – should be de/incremented carefully (in steps of one). |

| Number | Axis Parameter | Description |
|---|---|---|
| 193 | referencing mode | 1 – Only the left reference switch is searched.<br>2 – The right switch is searched and afterwards the left switch is searched.<br>3 – Three-switch-mode: the right switch is searched first and afterwards the reference switch will be searched.<br>Please see chapter 6.7.13 for details on reference search. |
| 194 | referencing search speed | For the reference search this value specifies the search speed as a fraction of the maximum velocity:<br>0 – full speed<br>1 – half of the maximum speed<br>2 – a quarter of the maximum speed<br>3 – 1/8 of the maximum speed (etc.) |
| 195 | referencing switch speed | Similar to parameter no. 194, the speed for the switching point calibration can be selected. |
| 203 | mixed decay threshold | If the actual velocity is above this threshold, mixed decay will be used. This can also be set to -1 which turns on mixed decay permanently also in the rising part of the microstep wave. This can be used to fix microstep errors. |
| 204 | freewheeling | Time after which the power to the motor will be cut when its velocity has reached zero. |
| 205 | stall detection threshold | Stall detection threshold. Set it to 0 for no stall detection or to a value between 1 (low threshold) and 7 (high threshold). The motor will be stopped if the load value exceeds the stall detection threshold. Switch off mixed decay to get usable results. |
| 211 | fullstep threshold | When exceeding this speed the driver will switch to real full step mode. To disable this feature set this parameter to zero or to a value greater than 2047.<br>Setting a full step threshold allows higher motor torque of the motor at higher velocity. When experimenting with this in a given application, try to reduce the motor current in order to be able to reach a higher motor velocity! |
| 214 | power down delay | Standstill period before the current is changed down to standby current. The standard value is 200 (value equates 2000msec). |

**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 |

## 6.7.29 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 8 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:**

| Number | Global parameter | Description |
|---|---|---|
| 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. |
| 65 | RS232 baud rate | 0 9600 baud *Default*<br>1 14400 baud<br>2 19200 baud<br>3 28800 baud<br>4 38400 baud<br>5 57600 baud<br>6 76800 baud *Not supported by Windows!*<br>7 (115200 baud) *3.68% Error (111111 Bits/s)* |
| 66 | serial address | The module (target) address for RS-232. |
| 67 | ASCII mode | Configure the TMCL™ ASCII interface:<br>Bit 0: 0 – start up in binary (normal) mode<br>1 – start up in ASCII mode<br>Bits 4 and 5:<br>00 – Echo back each character<br>01 – Echo back complete command<br>10 – Do not send echo, only send command reply |
| 73 | configuration EEPROM lock flag | Write: 1234 to lock the EEPROM, 4321 to unlock it.<br>Read: 1=EEPROM locked, 0=EEPROM unlocked. |
| 75 | telegram pause time | Pause time before the reply via RS232 is sent. For RS232 set to 0. |
| 76 | serial host address | Host address used in the reply telegrams sent back via RS232. |
| 77 | auto start mode | 0: Do not start TMCL™ application after power up (default).<br>1: Start TMCL™ application automatically after power up. |
| 80 | shutdown pin functionality | Select the functionality of the SHUTDOWN pin<br>0 – no function<br>1 – high active<br>2 – low active |

| Number | Global parameter | Description |
|---|---|---|
| 81 | TMCL™ code protection | Protect a TMCL™ program against disassembling or overwriting.<br>0 – no protection<br>1 – protection against disassembling<br>2 – protection against overwriting<br>3 – protection against disassembling and overwriting<br>***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.*** |
| 132 | tick timer | A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value. |

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

The global parameter bank 1 is normally not available. It may be used for customer specific extensions of the firmware. Together with user definable commands (see section 7.3) these variables form the interface between extensions of the firmware (written in C) and TMCL™ applications.

**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 |
|---|---|---|---|
| 0 | general purpose variable #0 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 1 | general purpose variable #1 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 2 | general purpose variable #2 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 3 | general purpose variable #3 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 4 | general purpose variable #4 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 5 | general purpose variable #5 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 6 | general purpose variable #6 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 7 | general purpose variable #7 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 8 | general purpose variable #8 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 9 | general purpose variable #9 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 10 | general purpose variable #10 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 11 | general purpose variable #11 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 12 | general purpose variable #12 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 13 | general purpose variable #13 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 14 | general purpose variable #14 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 15 | general purpose variable #15 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 16 | general purpose variable #16 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 17 | general purpose variable #17 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 18 | general purpose variable #18 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 19 | general purpose variable #19 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |
| 20..55 | general purpose variables #20..#55 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ |

***Please refer to chapter 8 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 |

## 6.7.30 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)
- EPO: clear the position error flag (modules with encoder feedback only)

**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<br>1 – (ETO) timeout flag<br>2 – (EAL) alarm flag<br>3 – (EDV) deviation flag<br>4 – (EPO) position flag<br>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 |
|---|---|---|---|---|---|---|---|---|---|
| **Function** | Target-address | Instruction Number | Type | Motor/ Bank | Operand Byte3 | Operand Byte2 | Operand Byte1 | Operand Byte0 | Checksum |
| **Value (hex)** | $01 | $24 | $01 | $00 | $00 | $00 | $00 | $00 | $26 |

### 6.7.31 Customer specific TMCL™ command extension (UF0...UF7/user function)

The user definable functions UF0... UF7 are predefined, functions without topic 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:** UF0... 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 |
|---|---|---|---|---|---|---|---|---|---|
| **Function** | Target-address | Target-address | Status | Instruction | Operand Byte3 | Operand Byte2 | Operand Byte1 | Operand Byte0 | Checksum |
| **Value (hex)** | $02 | $01 | (user defined) | 64...71 | (user defined) | (user defined) | (user defined) | (user defined) | <checksum> |

## 6.7.32 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) | 1 or 0 |

The *value* field contains a bit mask where every bit stands for one motor:

bit 0 = motor 0 (as only one motor is involved.)

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

| 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 | 100 | 138 | $00 | $00 | $00 | Motor bit mask | <checksum> |

The additional reply will be sent when all chosen motors have reached their target positions.

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

| 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 | 128 | 138 | $00 | $00 | $00 | Motor bit mask | <checksum> |

### 6.7.33 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.

## 6.7.34 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)
- *run application* (command 129)

***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<br>1 - run from specified address | (don't care) | (don't care)<br><br>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:<br>0 – stop<br>1 – run<br>2 – step<br>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<br>1 – binary | (don't care) | (don't care) |
| 137 – restore factory settings | reset all settings stored in the EEPROM to their factory defaults<br>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 6.6) | (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!

**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) |

# 7 Axis parameters

The following sections describe 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

## 7.1 Axis parameters

| Number | Axis Parameter | Description | Range [Unit] | Acc. |
|---|---|---|---|---|
| 0 | target (next) position | The desired position in position mode (see ramp mode, no. 138). | $\pm 2^{23}$ [µsteps] | RW |
| 1 | actual position | The current position of the motor. Should only be overwritten for reference point setting. | $\pm 2^{23}$ [µsteps] | RW |
| 2 | target (next) speed | The desired speed in velocity mode (see ramp mode, no. 138). In position mode, this parameter is set by hardware: to the maximum speed during acceleration, and to zero during deceleration and rest. | ±2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ | RW |
| 3 | actual speed | The current rotation speed. | ±2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ | RW |
| 4 | maximum positioning speed | Should not exceed the physically highest possible value. Adjust the pulse divisor (no. 154), if the speed value is very low (<50) or above the upper limit. See TMC 428 datasheet for calculation of physical units. | 0… 2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ | RWE |
| 5 | maximum acceleration | The limit for acceleration (and deceleration). Changing this parameter requires re-calculation of the acceleration factor (no. 146) and the acceleration divisor (no. 137), which is done automatically. See TMC 428 datasheet for calculation of physical units. | 0… 2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ | RWE |
| 6 | absolute max. current | The most important motor setting, since too high values might cause motor damage! The maximum value is 255 (which mean 100% of the maximum current of the module). | 0…255 [mA] | RWE |
| 7 | standby current | The current limit two seconds after the motor has stopped. | 0…255 [mA] | RWE |
| 8 | target pos. reached | Indicates that the actual position equals the target position. | 0/1 | R |
| 9 | ref. switch status | The logical state of the reference (left) switch. See the TMC 428 data sheet for the different switch modes. The default has two switch modes: the left switch as the reference switch, the right switch as a limit (stop) switch. | 0/1 | R |
| 10 | right limit switch status | The logical state of the (right) limit switch. | 0/1 | R |
| 11 | left limit switch status | The logical state of the left limit switch (in three switch mode) | 0/1 | R |
| 12 | right limit switch disable | If set, deactivates the stop function of the right switch | 0/1 | RWE |

| Number | Axis Parameter | Description | Range [Unit] | Acc. |
|---|---|---|---|---|
| 13 | left limit switch disable | Deactivates the stop function of the left switch resp. reference switch if set. | 0/1 | RWE |
| 130 | minimum speed | Should always be set 1 to ensure exact reaching of the target position. Do not change! | 0…2047 $\left[\frac{16\text{MHz}}{65536} \cdot 2^{PD} \frac{\mu\text{steps}}{\text{sec}}\right]$ | RWE |
| 135 | actual acceleration | The current acceleration (read only). | 0… 2047* | R |
| 138 | ramp mode | Automatically set when using ROR, ROL, MST and MVP.<br>0: position mode. Steps are generated, when the parameters actual position and target position differ. Trapezoidal speed ramps are provided.<br>2: velocity mode. The motor will run continuously and the speed will be changed with constant (maximum) acceleration, if the parameter *target speed* is changed.<br>For special purposes, the soft mode (value 1) with exponential decrease of speed can be selected. | 0/1/2 | RWE |
| 140 | microstep resolution | 0 – full step*)<br>1 – half step*)<br>2 – 4 microsteps<br>3 – 8 microsteps<br>4 – 16 microsteps<br>5 – 32 microsteps**)<br>6 – 64 microsteps**)<br>Note that modifying this parameter will affect the rotation speed in the same relation:<br>*) The full-step setting and the half-step setting are not optimized for use without an adapted microstepping table. These settings just step through the microstep table in steps of 64 respectively 32. To get real full stepping use axis parameter 211 or load an adapted microstepping table.<br>**) If the module is specified for 16 microsteps only, switching to 32 or 64 microsteps brings an enhancement in resolution and smoothness. The position counter will use the full resolution, but, however, the motor will resolve a maximum of 24 different microsteps only for the 32 or 64 microstep units. | 0… 6 | RWE |
| 141 | ref. switch tolerance | For three-switch mode: a position range, where an additional switch (connected to the REFL input) won't cause motor stop. See section 9.1 for details. | 0… 4095 | RW |
| 149 | soft stop flag | If cleared, the motor will stop immediately (disregarding motor limits), when the reference or limit switch is hit. | 0/1 | RWE |
| 153 | ramp divisor | The exponent of the scaling factor for the ramp generator- should be de/incremented carefully (in steps of one). | 0… 13 | RWE |

<table>
<tr><th>Number</th><th>Axis Parameter</th><th>Description</th><th>Range [Unit]</th><th>Acc.</th></tr>
<tr><td>154</td><td>pulse divisor</td><td>The exponent of the scaling factor for the pulse (step) generator – should be de/incremented carefully (in steps of one).</td><td>0… 13</td><td>RWE</td></tr>
<tr><td>193</td><td>referencing mode</td><td>1 – Only the left reference switch is searched.<br>2 – The right switch is searched and afterwards the left switch is searched.<br>3 – Three-switch-mode: the right switch is searched first and afterwards the reference switch will be searched.<br>Please see chapter 6.7.13 for details on reference search.</td><td>1/2/3</td><td>RWE</td></tr>
<tr><td>194</td><td>referencing search speed</td><td>For the reference search this value specifies the search speed as a fraction of the maximum velocity:<br>0 – full speed<br>1 – half of the maximum speed<br>2 – a quarter of the maximum speed<br>3 – 1/8 of the maximum speed (etc.)</td><td>0… 8</td><td>RWE</td></tr>
<tr><td>195</td><td>referencing switch speed</td><td>Similar to parameter no. 194, the speed for the switching point calibration can be selected.</td><td>0… 8</td><td>RWE</td></tr>
<tr><td>203</td><td>mixed decay threshold</td><td>If the actual velocity is above this threshold, mixed decay will be used. This can also be set to -1 which turns on mixed decay permanently also in the rising part of the microstep wave. This can be used to fix microstep errors.</td><td>0… 2048<br>or -1<br>$\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$</td><td>RWE</td></tr>
<tr><td>204</td><td>freewheeling</td><td>Time after which the power to the motor will be cut when its velocity has reached zero.</td><td>0… 65535<br>0 = never<br>[msec]</td><td>RWE</td></tr>
<tr><td>205</td><td>stall detection threshold</td><td>Stall detection threshold. Set it to 0 for no stall detection or to a value between 1 (low threshold) and 7 (high threshold). The motor will be stopped if the load value exceeds the stall detection threshold. Switch off mixed decay to get usable results.</td><td>0… 7</td><td>RWE</td></tr>
<tr><td>206</td><td>actual load value</td><td>Readout of the actual load value used for stall detection.</td><td>0… 7</td><td>R</td></tr>
<tr><td>208</td><td>Driver Error Flags of TMC249</td><td>
<table>
<tr><th>Bit</th><th>Name</th><th>Function</th><th>Remark</th></tr>
<tr><td>7</td><td>OT</td><td>Overtemperature</td><td>1 = chip of due to overtemperature</td></tr>
<tr><td>6</td><td>OTPW</td><td>Temperature prewarning</td><td>1= prewarning temperature exceeded</td></tr>
<tr><td>5</td><td>UV</td><td>Driver undervoltage</td><td>1 = undervoltage on VS</td></tr>
<tr><td>4</td><td>OCHS</td><td>Overcurrent high side</td><td>3 PWM cycles with overcurrent within 63 PWM cycles</td></tr>
<tr><td>3</td><td>OLB</td><td>Open load bridge B</td><td>No PWM switch off for 14 oscillator cycles</td></tr>
<tr><td>2</td><td>OLA</td><td>Open load bridge A</td><td>No PWM switch off for 14 oscillator cycles</td></tr>
<tr><td>1</td><td>OCB</td><td>Overcurrent bridge B low side</td><td>3 PWM cycles with overcurrent within 63 PWM cycles</td></tr>
<tr><td>0</td><td>OCA</td><td>Overcurrent bridge A low side</td><td>3 PWM cycles with overcurrent within 63 PWM cycles</td></tr>
</table>
</td><td>0… 7</td><td>R</td></tr>
</table>

| Number | Axis Parameter | Description | Range [Unit] | Acc. |
|---|---|---|---|---|
| 211 | fullstep threshold | When exceeding this speed the driver will switch to real full step mode. To disable this feature set this parameter to zero or to a value greater than 2047.<br>Setting a full step threshold allows higher motor torque of the motor at higher velocity. When experimenting with this in a given application, try to reduce the motor current in order to be able to reach a higher motor velocity! | 0..2048<br>$\left[\frac{16\text{MHz}}{65536} \cdot 2^{\text{PD}} \frac{\mu\text{steps}}{\text{sec}}\right]$ | RWE |
| 214 | power down delay | Standstill period before the current is changed down to standby current. The standard value is 200 (value equates 2000msec). | 1... 65535 [10msec] | RWE |

* Unit of acceleration: $\frac{16MHz^2}{536870912 \cdot 2^{puls\_divisor+ramp\_divisor}} \frac{\text{microsteps}}{\text{sec}^2}$

***Please use the TMCL-IDE axis parameter calculation tool for getting best values.***

# 8 Global parameters

The global parameters apply for all types of TMCM modules.

**They are grouped into 3 banks:**

- bank 0 (global configuration of the module)
- bank 1 (normally not available; for customer specific extensions of the firmware)
- bank 2 (user TMCL™ variables)

Please use SGP and GGP commands to write and read global parameters. Further you can use the STGP command in order to store TMCL™ user variables permanently in the EEPROM of the module. With the RSGP command the contents of a user variable can be restored from the EEPROM, if this is necessary.

## 8.1 Bank 0

**Parameters 0...38**

The first parameters 0...38 are only mentioned here for completeness. They are used for the internal handling of the TMCL-IDE and serve for loading micro step and driver tables. Normally these parameters remain untouched. ***If you want to use them for loading your specific values with your PC software please contact TRINAMIC and ask how to do this. Otherwise you might cause damage on the motor driver!***

| Number | Parameter |
|---|---|
| 0 | datagram low word (read only) |
| 1 | datagram high word (read only) |
| 2 | cover datagram position |
| 3 | cover datagram length |
| 4 | cover datagram contents |
| 5 | reference switch states (read only) |
| 6 | TMC428 SMGP register |
| 7...22 | driver chain configuration long words 0...15 |
| 23...38 | microstep table long word 0...15 |

**Parameters 64...132**

Parameters with numbers from 64 on configure stuff like the serial address of the module RS232 baud 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. A 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.***

**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 |

| Number | Global parameter | Description | | | Range | Access |
|---|---|---|---|---|---|---|
| 65 | RS232 baud rate | 0 | 9600 baud | *Default* | 0…7 | RWE |
| | | 1 | 14400 baud | | | |
| | | 2 | 19200 baud | | | |
| | | 3 | 28800 baud | | | |
| | | 4 | 38400 baud | | | |
| | | 5 | 57600 baud | | | |
| | | 6 | 76800 baud | *Not supported by Windows!* | | |
| | | 7 | (115200 baud) | *3.68% Error (111111 Bits/s)* | | |
| 66 | serial address | The module (target) address for RS-232. | | | 0…255 | RWE |
| 67 | ASCII mode | Configure the TMCL™ ASCII interface:<br>Bit 0: 0 – start up in binary (normal) mode<br>1 – start up in ASCII mode<br>Bits 4 and 5:<br>00 – Echo back each character<br>01 – Echo back complete command<br>10 – Do not send echo, only send command reply | | | | RWE |
| 73 | configuration EEPROM lock flag | Write: 1234 to lock the EEPROM, 4321 to unlock it.<br>Read: 1=EEPROM locked, 0=EEPROM unlocked. | | | 0/1 | RWE |
| 75 | telegram pause time | Pause time before the reply via RS232 is sent. For RS232 set to 0. | | | 0…255 | RWE |
| 76 | serial host address | Host address used in the reply telegrams sent back via RS232. | | | 0…255 | RWE |
| 77 | auto start mode | 0: Do not start TMCL™ application after power up (default).<br>1: Start TMCL™ application automatically after power up. | | | 0/1 | RWE |
| 80 | shutdown pin functionality | Select the functionality of the SHUTDOWN pin<br>0 – no function<br>1 – high active<br>2 – low active | | | 0…2 | RWE |
| 81 | TMCL™ code protection | Protect a TMCL™ program against disassembling or overwriting.<br>0 – no protection<br>1 – protection against disassembling<br>2 – protection against overwriting<br>3 – protection against disassembling and overwriting<br>***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.*** | | | 0,1,2,3 | RWE |
| 128 | TMCL™ application status | 0 –stop<br>1 – run<br>2 – step<br>3 – reset | | | 0…3 | R |
| 129 | download mode | 0 – normal mode<br>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 |
| 133 | random number | Choose a random number. ***Read only!*** | | | 0…2147483647 | R |

## 8.2 Bank 1

The global parameter bank 1 is normally not available. It may be used for customer specific extensions of the firmware. Together with user definable commands (see section 7.3) these variables form the interface between extensions of the firmware (written in C) and TMCL™ applications.

## 8.3 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.

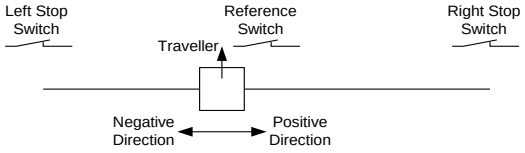
| Number | Global parameter | Description | Range | Access |
|---|---|---|---|---|
| 0 | general purpose variable #0 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ | RWE |
| 1 | general purpose variable #1 | for use in TMCL™ applications | $-2^{31}...+2^{31}$ | RWE |
| 2 | general purpose variable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

# 9 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.

## 9.1 Reference search

The built-in reference search features switching point calibration and supports of one reference switch per axis. 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) do not have any 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 Figure 9.1 the connection of the left and the right limit switch is shown. Figure 9.2 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.

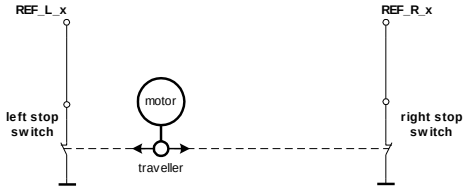



**Figure 9.1: Two limit switches**

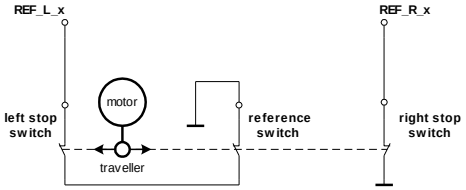



**Figure 9.2: Limit switches with extra reference switch**

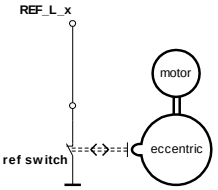



**Figure 9.3: Circular system**

## 9.2 Stall detection

The TMCM-109 is offered with stallGuard™. In this case the module is equipped with a TMC249 motor driver chip, which features load measurement that can be used for stall detection. Stall detection means that the motor will be stopped when the load gets too high. It is controlled by axis parameter #205. If this parameter is set to a value between 1 and 7 the stall detection will be activated. Setting it to 0 means that stall detection is turned off. A greater value means a higher threshold. This also depends on the motor and on the velocity. There is no stall detection while the motor is being accelerated or decelerated.

Stall detection can also be used for finding the reference point. You can do this by using the following TMCL™ code:

```
      SAP 205, 0, 5   //Turn on Stall Detection (use other threshold if needed)
      ROL 0, 500      //Let the motor run (or use ROR or other velocity)
Loop: GAP 3, 0
      COMP 0
      JC NE, Loop     //Wait until the motor has stopped
      SAP 1, 0, 0     //Set this position as the zero position
```

***Do not use RFS in this case.***

***Mixed decay should be switched off when stallGuard™ is operational in order to get usable results.***

## 9.3 Fixing microstep errors

Due to the *zero crossing problem* of the TMC249 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 threshold* parameter (axis parameter number 203) to the value -1. This switches on mixed decay permanently, in every part of the microstepping waveform. Now the minimum reachable motor current is always near zero which gives better microstepping results.
A further optimization is possible by adapting the motor current shape. (For further information about TMCL-IDE please refer to the TMCL™ reference and programming manual.)

*Use **SAP 203, <motor number>, -1** to turn on this feature.*

# 10 Revision history

## 10.1 Firmware revision

| Version | Date | Author | Description |
|---|---|---|---|
| 3.37 | 2009-JAN-29 | OK | |
| | | | |

## 10.2 Document revision

| Version | Date | Author | Description |
|---|---|---|---|
| 1.00 | 2010-FEB-18 | SD | Initial version |
| | | | |

# 11 References

[PDx-109-57 V2] PDx-109-57 V2 Hardware Manual (see http://www.trinamic.com)