Firmware Version V1.14

TMCL™ FIRMWARE MANUAL

+ +

+ +

TMCM-1311

1-Axis Stepper
Closed Loop Controller/Driver
3.0 A / 48 V
Encoder Input
18 GPIOs
USB, CAN, RS485

TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

www.trinamic.com

Table of Contents

2 Features

The TMCM-1311 is a single axis stepper motor controller and driver standalone board with USB, CAN and RS485 interface. It supports motor currents up to 3A RMS and supply voltages up to 48V nominal. The module offers inputs for an incremental a/b/n (TTL, open-collector, and differential inputs) encoder. In addition, there are dedicated stop switch inputs and 8 general purpose inputs and 8 general purpose outputs.

Main Characteristics

Bipolar stepper motor driver

- Up to 256 microsteps per full step
- High-efficient operation, low power dissipation
- Dynamic current control
- Integrated protection: overtemperature and undervoltage
- stallGuard2™ feature for stall detection (for open load operation)

Encoder

- Encoder input for incremental a/b/n (TTL, OC or diff.)

Interfaces

- RS485 2-wire communication interface
- CAN 2.0B communication interface (4x DIP-switch for CAN / RS485 address setting or other settings)
- USB 2.0 full-speed (12Mbit/s) communication interface (mini-USB connector)
- Encoder input: incremental a/b/n (TTL, OC or diff.)
- Dedicated STOP_L / STOP_R inputs
- Up to 8 multi-purpose inputs (+24V compatible, incl. 2 dedicated analog inputs)
- Up to 8 multi-purpose outputs (Open-drain, incl. 2 outputs for currents up to 1A)

Software

- TMCL™ remote (direct mode) and standalone operation with memory for up to 1024 TMCL commands
- Closed-loop support
- Fully supported by TMCL-IDE (PC based integrated development environment)

Electrical data

- Supply voltage: +12V… +48V DC
- Motor current: up to 3A RMS (programmable)

Mechanical data

- Board size: 110mm x 110mm, height 26.3mm max.

Please refer to separate TMCM-1311 Hardware Manual for additional information.

TRINAMIC Features – Closed Loop Mode

The TMCM-1311 is mainly designed to run 2-phase stepper motors in closed loop mode. It offers an automatic motor load adaption in positioning mode, velocity mode, and torque mode, which is based on encoder feedback and closed loop control software for analysis, error detection and error correction.

The closed loop mode operation combines the advantages of a stepper driver system with the benefits of a servo drive. Thus, the TMCM-1311 is able to satisfy ambitious requirements in reliability and precision and can be used in several industrial demanding applications.

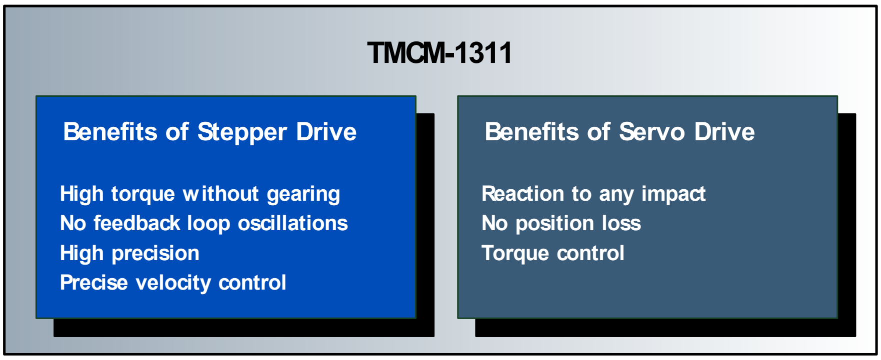


Figure 2.1 TMCM-1311 characteristics in closed loop mode

The TRINAMIC Closed Loop Mode operation

- prevents the motor from stall and step loss caused by too high load or high velocity.
- adapts the current amplitude to each motor load which is within the ranges predetermined by motor and controller/driver board characteristics.
- achieves a higher torque output than in open loop mode.
- guarantees a precise and fast positioning.
- enables velocity and positioning error compensation.

Using the TMCM-1311, energy will be saved and the motor will be kept cool.

3 Putting the TMCM-1311 into Operation

Here you can find basic information for putting your TMCM-1311 into operation. If you are already common with TRINAMICs modules you may skip this chapter.

THE THINGS YOU NEED:

- TMCM-1311
- Interface (RS485/CAN/USB) suitable to your module with cables
- Nominal supply voltage +24V DC or +48V DC for your module
- TMCM-1310/1311-GUI parameterization software, TMCL-IDE program, and PC
- Stepper motor
- Encoder (in case, closed loop mode operation is desired)

PRECAUTIONS
Do not connect or disconnect the TMCM-1311 while powered!
Do not connect or disconnect the motor while powered!
Do not exceed the maximum power supply voltage of 48V DC!
Note, that the module is not protected against reverse polarity!
START WITH POWER SUPPLY OFF!

3.1 Connecting the TMCM-1311

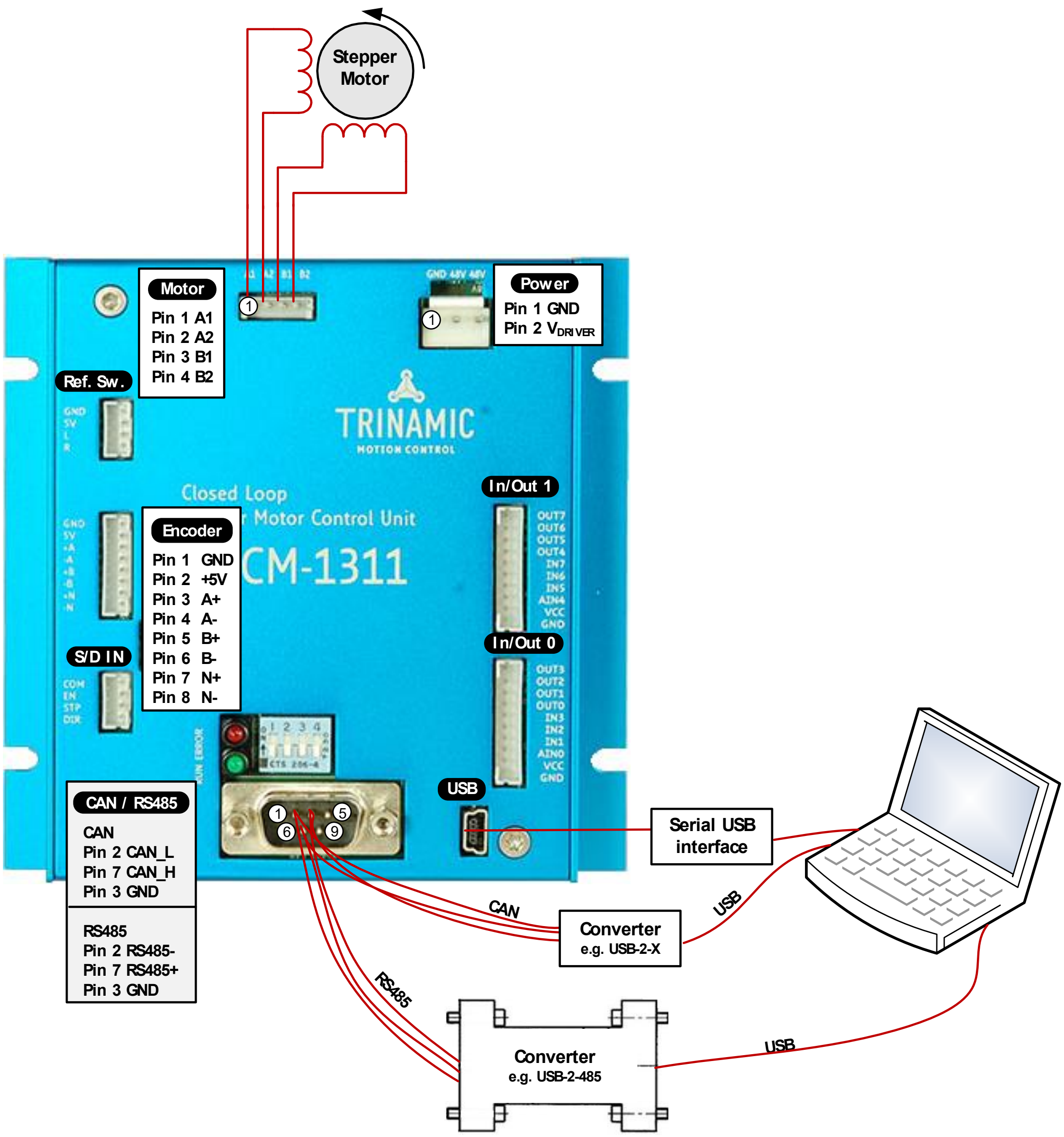


Figure 3.1 Starting up

1. **Choose your interface**
According to default settings of the module, USB interface and CAN interface can be used without any further configuration. For operation with RS485 interface, change the interface setting (global parameter 88) first.

a) CAN interface

Pin	Label	Direction	Description
1			
2	CAN_L	Bi-directional	Differential CAN bus signal (inverting)
3	GND	Power (GND)	Signal and system ground
4			
5			
6			
7	CAN_H	Bi-directional	Differential CAN bus signal (non-inverting)
8			
9			

b) RS485 interface
Before connecting the interface it is necessary to configure it. For this purpose, use the USB interface and the software development environment TMCL-IDE. Using global parameter 88 (*interface selection*) the RS485 interface can be selected. This setting will automatically be permanently stored in the EEPROM of the module.

TMCL-command for changing to the RS485 interface: *SGP 88, 0, 1.*

Pin	Label	Direction	Description
1			
2	RS485-	Bi-directional	Differential RS485 bus signal (inverting)
3	GND	Power (GND)	Signal and system ground
4			
5			
6			
7	RS485+	Bi-directional	Differential RS485 bus signal (non-inverting)
8			
9			

c) USB interface
For using USB interface download and install the file *TMCM-1311.inf* (www.trinamic.com).

Pin	Label	Direction	Description
1	VBUS	Power (+5V input)	+5V supply from Host
2	D-	Bi-directional	USB Data -
3	D+	Bi-directional	USB Data +
4	ID		Connected to signal and system ground
5	GND	Power (GND)	Signal and System ground

2. **Connect the motor**

Pin	Label	Direction	Description
1	A1	Output	Pin 1 of motor coil A
2	A2	Output	Pin 2 of motor coil A
3	B1	Output	Pin 1 of motor coil B
4	B2	Output	Pin 2 of motor coil B

3. Connect the encoder (*optional - for closed loop operation*)

Pin	Label	Direction	Description
1	GND	Power (GND)	Signal and system ground
2	+5V	Power (supply output)	+5V output for external circuit
3	A+	Input	Encoder channel A+ input (differential, non-inverting)
4	A-	Input	Encoder channel A- input (differential, inverting)
5	B+	Input	Encoder channel B+ input (differential, non-inverting)
6	B-	Input	Encoder channel B- input (differential, inverting)
7	N+	Input	Encoder zero / index channel input (differential, non-inverting)
8	N-	Input	Encoder zero / index channel input (differential, inverting)

4. Connect the power supply

Pin	Label	Direction	Description
1	GND	Power (GND)	Common system supply and signal ground
2	V_{DRIVER}	Power (supply input)	Stepper driver supply voltage. Without this voltage the stepper driver and any motor connected will not be energized.
3	$V_{DIGITAL}$	Power (supply input)	Supply voltage for everything else apart from the stepper motor driver. The on-board voltage regulator generates the necessary voltages for the digital circuits from this supply. The pin can be left unconnected. In this case a diode between V_{DRIVER} and $V_{DIGITAL}$ ensures the supply for the digital parts. *ATTENTION:* - *The diode has a current rating of 3A. As $V_{DIGTIAL}$ is available at the I/O connectors and at the reference switch connectors also, always connect this pin to positive supply voltage in case substantial amount of current is withdrawn from these pins for external circuits.* - *It is expected that $V_{DIGITAL}$ and V_{DRIVER} are connected to the same power supply output when both pins are used. Otherwise please ensure that $V_{DIGITAL}$ is always equal or higher than V_{DRIVER} when connected (due to the diode).*

5. Switch ON the power supply

Turn power ON. The green LED should be flashing. The motor is powered but in standstill now.

If this does not occur, switch power OFF and check your connections as well as the power supply.

Refer to the hardware manual for further information about the hardware characteristics of your module!

3.2 Finding Optimum Settings with Parameterization Tool

Evaluating best parameter values for the TMCM-1311 with the TMCM-1310/1311-GUI software tool is quite easy. The TMCM-1310/1311-GUI is available on www.trinamic.com.

After connecting the module as described in chapter 3.1, start the TMCM-1310/1311-GUI with a double click. The following window will appear on the screen (see Figure 3.2).

The software detects your module and the related virtual com port for USB automatically if the file *TMCM-1311.inf* is installed correctly. Now, click ***Connect*** to connect the module and start the trace controller in case you like to work with graphs for velocity, positioning, or torque.

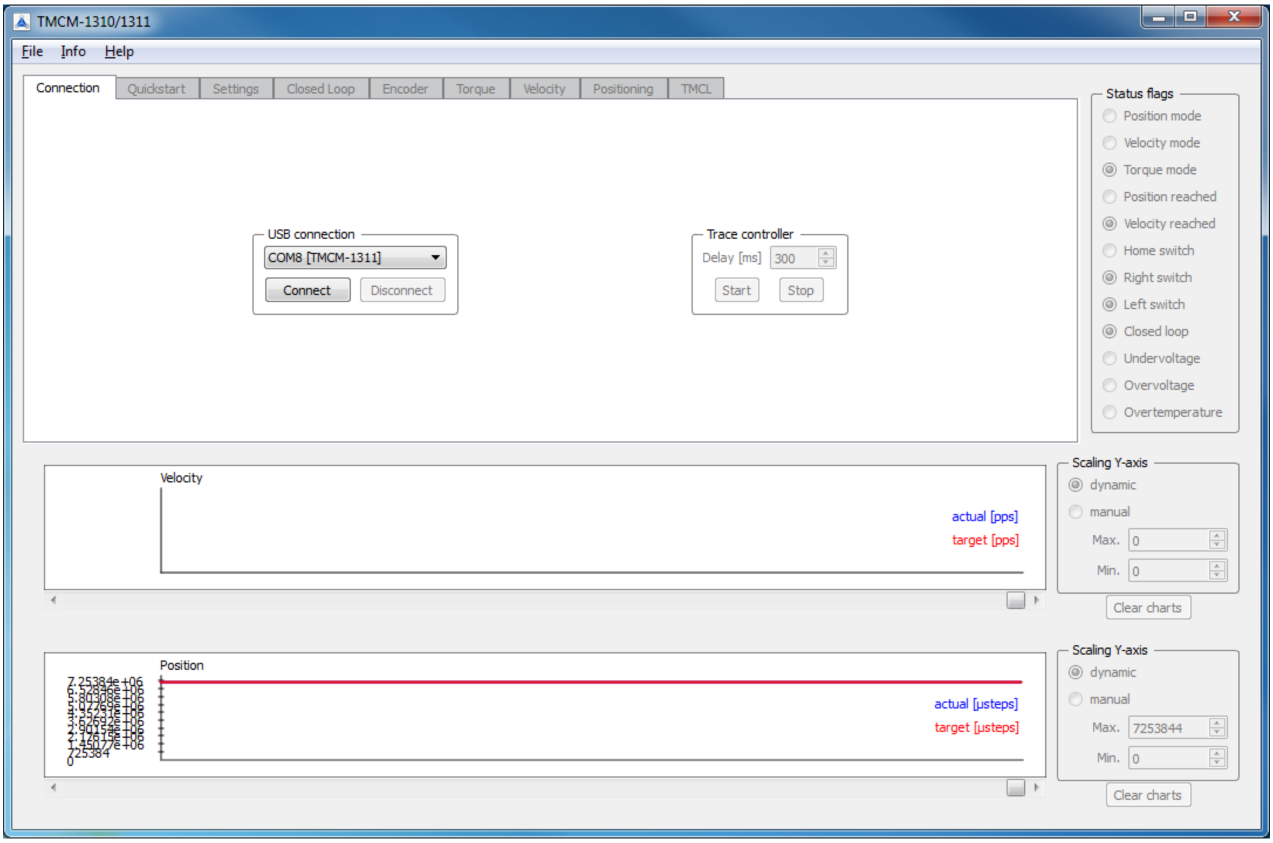


Figure 3.2 Welcome window of TMCM-1310/1311-GUI

The graphs for velocity, positioning, and torque can be selected via the *Info* menu as shown in Figure 3.3.

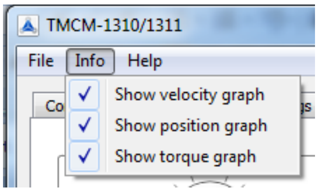


Figure 3.3 Info menu

Quickstart

Using the default values of the TMCM-1311, it is possible to proceed with a ***quickstart***. This way, you can check how the module drives your specific stepper motor using these default values. Calibrate them after first tests to find optimum values.

There are only two things that have to be done before the quickstart:

- For closed loop mode, initialize the encoder with the appropriate command fields (see **Figure 3.5**) first. To open up the *encoder initialization* window, click on the ***Init encoder*** button on the ***quickstart*** tab.
- Now, choose ***closed loop*** mode or ***open loop*** mode operation.

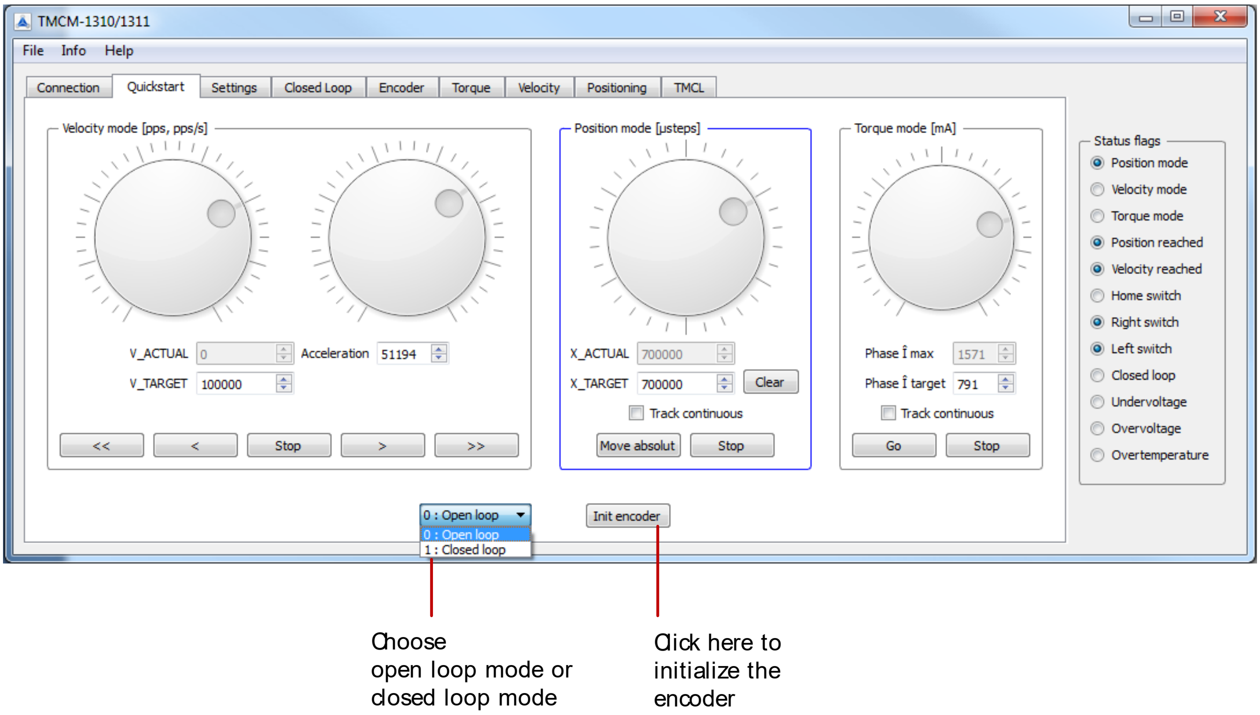


Figure 3.4 Quickstart tab

How to initialize the encoder

Fill in the ***fullstep resolution*** of your motor and click ***Start encoder initialization***. The motor will rotate just one round to initialize the encoder. If the encoder has been detected successfully, the appropriate initialization status will be shown. In case an error occurred, this will be shown in the status field also (state 3 = encoder detection failed). In this case check your hardware connections and the motor fullstep resolution.

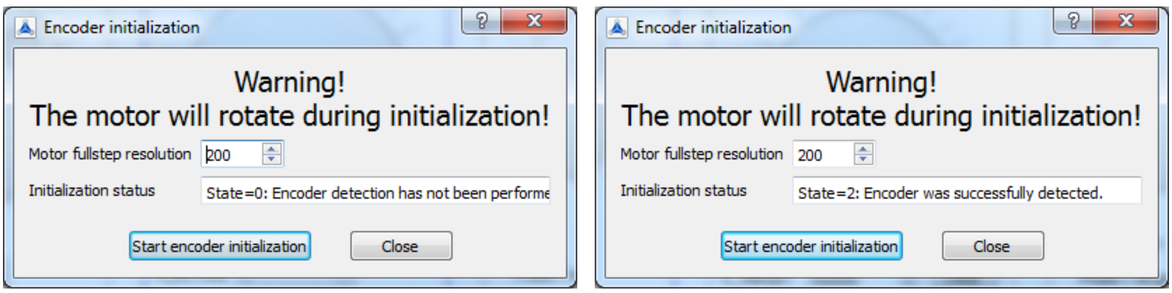


Figure 3.5 Encoder initialization window before initialization and afterwards

Now, first steps are made. Parameterize your module using the other tabs of the software tool.

3.3 Using the TMCL-IDE to Develop Programs

The TMCL-IDE is available 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 the parameters shown in Figure 3.6 (baud rate 9600). Click ***OK***.

USB interface
If the file *TMCM-1311.inf* is installed correctly, the module will be identified automatically.

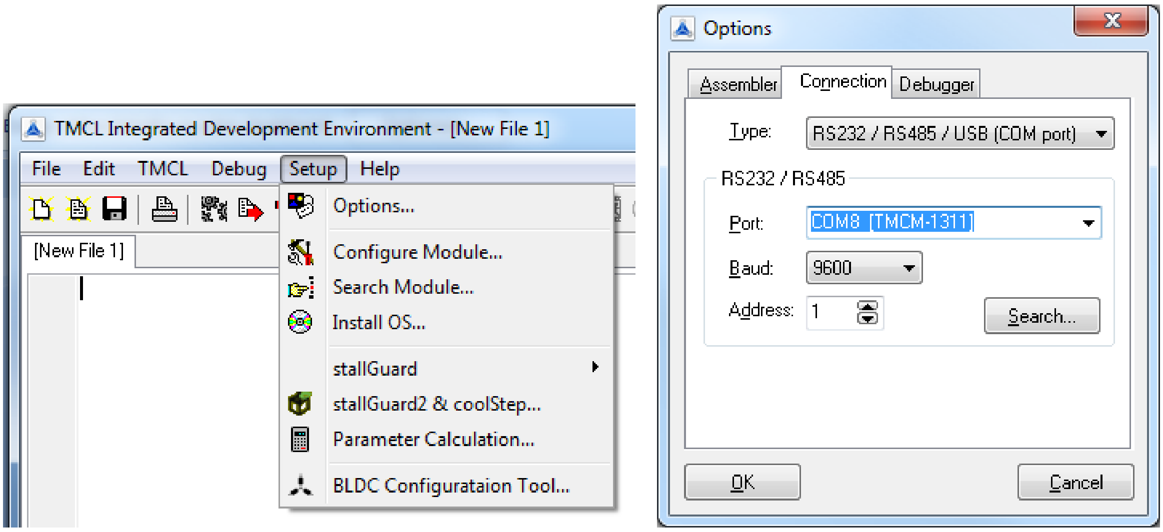
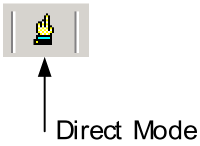


Figure 3.6 Setup dialogue and connection tab of the TMCL-IDE.

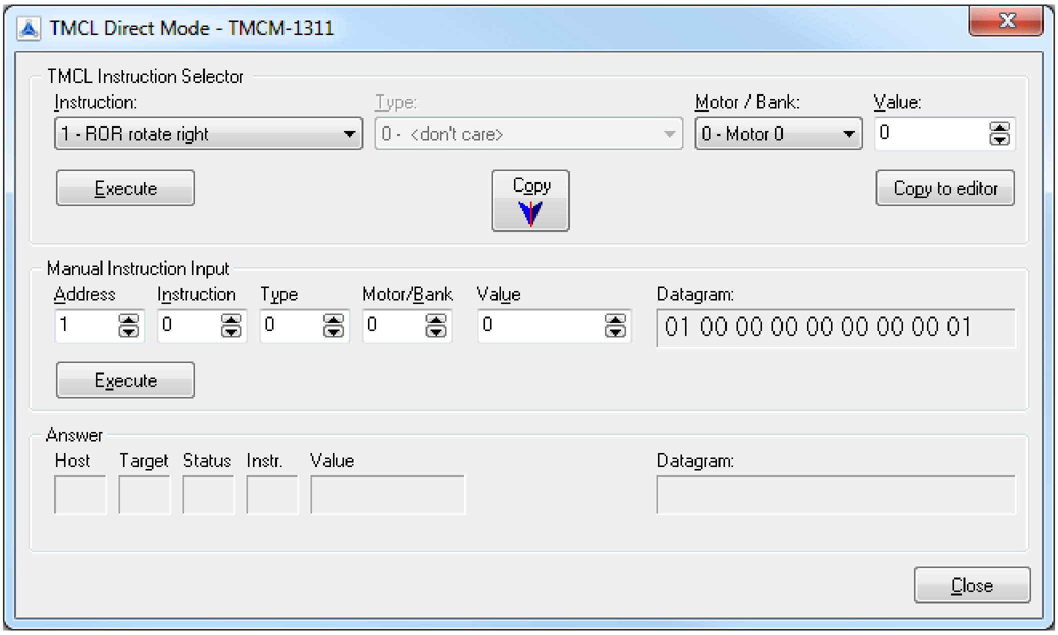
Please refer to the TMCL-IDE User Manual for more information (see www.TRINAMIC.com).

3.3.1 Using TMCL Direct Mode

1. Start TMCL ***Direct Mode***.



2. If the communication is established the TMCM-1311 will be detected automatically. *If the module should not be 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 50000 -> Click *Execute*. The motor is rotating now.
- MST motor stop, motor 0 -> Click *Execute*. The motor stops now.

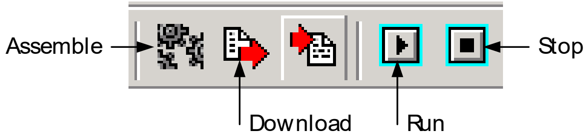
Top right of the ***TMCL Direct Mode*** window is the button ***Copy to editor***. Click here to copy the chosen command and create your own TMCL program. The command will be shown immediately on the editor.

3.3.2 Testing with a Simple TMCL Program

Type in the following program:

```
      ROL 0, 50000            //Rotate motor 0 with speed 50000
      WAIT TICKS, 0, 500
      MST 0
      ROR 0, 50000            //Rotate motor 0 with 50000
      WAIT TICKS, 0, 500
      MST 0

      SAP 4, 0, 50000         //Set max. Velocity
      SAP 5, 0, 50000         //Set max. Acceleration
Loop: MVP ABS, 0, 100000      //Move to Position 10000
      WAIT POS, 0, 0          //Wait until position reached
      MVP ABS, 0, -100000     //Move to Position -10000
      WAIT POS, 0, 0          //Wait until position reached
      JA Loop                 //Infinite Loop
```



1. Click the ***Assemble*** icon to convert the TMCL program into binary code.
2. Then download the program to the TMCM-1311 module by clicking the ***Download*** icon.
3. Click the ***Run*** icon. The desired program will be executed.
4. Click the ***Stop*** button to stop the program.

3.4 Important Motor Settings

There are some axis parameters which have to be adjusted right in the beginning after installing your module. Please set the upper limiting values for the speed (axis parameter 4), the acceleration (axis parameter 5), and the current (axis parameter 6). Further set the standby current (axis parameter 7) and choose your microstep resolution with axis parameter 140. Please use the *SAP* (Set Axis Parameter) command for adjusting these values. The SAP command is described in paragraph 5.7.5. You can use the TMCL-IDE direct mode for easily configuring your module.

> *Attention*:
> The most important motor setting is the *absolute maximum motor current* setting, since too high values might cause motor damage!

IMPORTANT AXIS PARAMETERS FOR MOTOR SETTING

Number	Axis Parameter	Description	Range [Unit]
4	Maximum positioning speed	Should not exceed the physically highest possible value. Adjust the pulse divisor (axis parameter 154), if the speed value is very low (<50) or above the upper limit.	0… +268.435.454 [pps]
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 429 datasheet for calculation of physical units.	1… +33554431 [pps/s]
6	Absolute max. current (CS / Current Scale)	The maximum value is 255. This value means 100% of the maximum current of the module. The current adjustment is within the range 0… 255 and can be adjusted in 32 steps. 0… 7 | 79…87 | 160… 167 | 240… 247 8… 15 | 88… 95 | 168… 175 | 248… 255 16… 23 | 96… 103 | 176… 183 24… 31 | 104… 111 | 184… 191 32… 39 | 112… 119 | 192… 199 40… 47 | 120… 127 | 200… 207 48… 55 | 128… 135 | 208… 215 56… 63 | 136… 143 | 216… 223 64… 71 | 144… 151 | 224… 231 72… 79 | 152… 159 | 232… 239 *The most important motor setting, since too high values might cause motor damage!*	0… 255 $I_{peak} =< value > \times \frac{4.2A}{255}$ $I_{RMS} =< value > \times \frac{3.0A}{255}$
7	Standby current	The current limit two seconds after the motor has stopped.	0… 255 $I_{peak} =< value > \times \frac{4.2A}{255}$ $I_{RMS} =< value > \times \frac{3.0A}{255}$
140	Microstep resolution	0 full step 1 half step 2 4 microsteps 3 8 microsteps 4 16 microsteps 5 32 microsteps 6 64 microsteps 7 128 microsteps 8 256 microsteps	0… 8

4 TMCL and the TMCL-IDE: Introduction

The software running on the microprocessor of the TMCM-1311 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 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 TMCM-1311 supports TMCL direct mode (binary commands) and standalone TMCL program execution. You can store up to 2048 TMCL instructions on it. In direct mode and most cases the TMCL communication over RS485, CAN, or USB follows a strict master/slave relationship. That is, a host computer (e.g. PC/PLC) acting as the interface bus master will send a command to the TMCM-1311. The TMCL interpreter on the module will then interpret this command, do the initialization of the motion controller, read inputs and write outputs or whatever is necessary according to the specified command. As soon as this step has been done, the module will send a reply back over RS485/CAN/USB to the bus master. Only then should the master transfer the next command. Normally, the module will just switch to transmission and occupy the bus for a reply, otherwise it will stay in receive mode. It will not send any data over the interface without receiving a command first. This way, any collision on the bus will be avoided when there are more than two nodes connected to a single bus.

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

Every command has a binary representation and a mnemonic. The binary format is used to send commands from the host to a module in direct mode, whereas the mnemonic format is used for easy usage of the commands when developing standalone TMCL applications using 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.

4.1 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 RS485 or USB interface, it has to be enclosed by an address byte at the beginning and a checksum byte at the end. In this case it consists of nine bytes.

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

The binary command format for R485/USB is as follows:

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

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

4.1.1 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
```

4.2 Reply Format

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

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

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

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

4.2.1 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

4.3 Standalone Applications

The module is equipped with a TMCL memory for storing TMCL applications. You can use TMCL-IDE for developing standalone TMCL applications. You can download a program into the EEPROM and afterwards it will run on the module. The TMCL-IDE contains an editor and the 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.

4.4 The ASCII Interface

There is also an ASCII interface that can be used to communicate with the module and to send some commands as text strings.

ENTERING AND LEAVING ASCII MODE:

- 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-entering the binary mode enter the command BIN.

4.4.1 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.

EXAMPLES FOR VALID COMMAND LINES

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

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

4.4.2 Format of a Reply

After executing the command the module sends back a reply in ASCII format. The 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*.

4.4.3 Commands 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, SCO, GCO, CCO, UF0, UF1, UF2, UF3, UF4, UF5, UF6, and UF7.

SPECIAL COMMANDS WHICH 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.

Note: Only direct mode commands can be entered in ASCII mode!

4.4.4 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 10.1).

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. This may be useful in RS485 systems.

5 TMCL Commands

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

5.1 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 standalone mode.

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

5.2 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 standalone mode.

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

5.3 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 standalone mode only.

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

5.4 I/O Port Commands

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

Mnemonic	Command number	Meaning
SIO	14	Set output
GIO	15	Get input

5.5 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
ACO	39	Copy accu to coordinate

For calculating purposes there is an accumulator (or accu or A register) and an X register. When executed in a TMCL program (in standalone 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 (standalone mode), a host can still send commands like GAP and GGP to the module (e.g. to query the actual position of the motor) without affecting the flow of the TMCL program running on the module.

5.6 Interrupt Commands

Due to some customer requests, interrupt processing has been introduced in the TMCL firmware for ARM based modules.

Mnemonic	Command number	Meaning
EI	25	Enable interrupt
DI	26	Disable interrupt
VECT	37	Set interrupt vector
RETI	38	Return from interrupt

5.6.1 Interrupt Types:

There are many different interrupts in TMCL, like timer interrupts, stop switch interrupts, position reached interrupts, and input pin change interrupts. Each of these interrupts has its own interrupt vector. Each interrupt vector is identified by its interrupt number. Please use the TMCL included file *Interrupts.inc* for symbolic constants of the interrupt numbers.

5.6.2 Interrupt Processing:

When an interrupt occurs and this interrupt is enabled and a valid interrupt vector has been defined for that interrupt, the normal TMCL program flow will be interrupted and the interrupt handling routine will be called. Before an interrupt handling routine gets called, the context of the normal program will be saved automatically (i.e. accumulator register, X register, TMCL flags).

There is no interrupt nesting, i.e. all other interrupts are disabled while an interrupt handling routine is being executed.

On return from an interrupt handling routine, the context of the normal program will automatically be restored and the execution of the normal program will be continued.

5.6.3 Interrupt Vectors:

The following table shows all interrupt vectors that can be used.

Interrupt number	Interrupt type
0	Timer 0
1	Timer 1
2	Timer 2
3	Target position reached
15	stallGuard2
21	Deviation
27	Left stop switch
28	Right stop switch
39	Input change 0
40	Input change 1
41	Input change 2
42	Input change 3
43	Input change 4
44	Input change 5
45	Input change 6
46	Input change 7
255	Global interrupts

5.6.4 Further Configuration of Interrupts

Some interrupts need further configuration (e.g. the timer interval of a timer interrupt). This can be done using SGP commands with parameter bank 3 (SGP <type>, 3, <value>). Please refer to the SGP command (paragraph 5.7.9) for further information about that.

5.6.5 Using Interrupts in TMCL

To use an interrupt the following things have to be done:

- Define an interrupt handling routine using the VECT command.
- If necessary, configure the interrupt using an SGP <type>, 3, <value> command.
- Enable the interrupt using an EI <interrupt> command.
- Globally enable interrupts using an EI 255 command.
- An interrupt handling routine must always end with a RETI command

EXAMPLE FOR THE USE OF A TIMER INTERRUPT:

```
   VECT 0, Timer0Irq  //define the interrupt vector
   SGP 0, 3, 1000     //configure the interrupt: set its period to 1000ms
   EI 0               //enable this interrupt
   EI 255             //globally switch on interrupt processing

//Main program: toggles output 3, using a WAIT command for the delay
Loop:
   SIO 3, 2, 1
   WAIT TICKS, 0, 50
   SIO 3, 2, 0
   WAIT TICKS, 0, 50
   JA Loop

//Here is the interrupt handling routine
Timer0Irq:
   GIO 0, 2          //check if OUT0 is high
   JC NZ, Out0Off    //jump if not
   SIO 0, 2, 1       //switch OUT0 high
   RETI              //end of interrupt
Out0Off:
   SIO 0, 2, 0       //switch OUT0 low
```

```
RETI        //end of interrupt
```

In this example the interrupt numbers are used directly. To make the program better readable use the provided include file *Interrupts.inc.* This file defines symbolic constants for all interrupt numbers which can be used in all interrupt commands. The beginning of the program above then looks like the following:

```
#include Interrupts.inc
  VECT TI_TIMER0, Timer0Irq
  SGP TI_TIMER0, 3, 1000
  EI TI_TIMER0
  EI TI_GLOBAL
```

Please also take a look at the other example programs.

5.6.6 ASCII Commands

Mnemonic	Command number	Meaning
-	139	Enter ASCII mode
BIN	-	Quit ASCII mode and return to binary mode. This command can only be used in ASCII mode.

5.7 Commands

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

5.7.1 ROR (rotate right)

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

Like on all other TMCL modules, the motor will be accelerated or decelerated to the speed given with the command. The speed is given in microsteps per second (pps). For conversion of this value into rounds per minute etc. please refer to chapter 0, also.

The range is -268.435.455… +268.435.454.

Internal function: first, velocity mode is selected. Then, the velocity value is transferred to axis parameter #2 (*target velocity*).

Related commands: ROL, MST, SAP, GAP

Mnemonic: ROR 0, <velocity>

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
1	don't care	<motor> 0*	<velocity> -268.435.455… +268.435.454

*** Motor number is always O as the module support just one axis**

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

Example:

Rotate right, velocity = 10000
Mnemonic: ROR 0, 10000

Binary:

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$01	$00	$00	$00	$00	$27	$10

5.7.2 ROL (rotate left)

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

Like on all other TMCL modules, the motor will be accelerated or decelerated to the speed given with the command. The speed is given in microsteps per second (pps). For conversion of this value into rounds per minute etc. please refer to chapter 0, also.

The range is -268.435.455… +268.435.454.

Internal function: first, velocity mode is selected. Then, the velocity value is transferred to axis parameter #2 (*target velocity*).

Related commands: ROR, MST, SAP, GAP

Mnemonic: ROL 0, <velocity>

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
2	don't care	<motor> 0*	<velocity> -268.435.455… +268.435.454

*** Motor number is always O as the module support just one axis**

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

Example:

Rotate left, velocity = 10000
Mnemonic: ROL 0, 10000

Binary:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$02	$00	$00	$00	$00	$27	$10

5.7.3 MST (motor stop)

The motor will be instructed to stop.

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

Related commands: ROL, ROR, SAP, GAP

Mnemonic: MST 0

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
3	don't care	<motor> 0*	don't care

*** Motor number is always O as the module support just one axis**

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

Example:

Stop motor 0
Mnemonic: MST 0

Binary:

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

5.7.4 MVP(move to position)

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.

UNITS AND RANGE

Open loop: the range of the MVP command is 32 bit signed (−2.147.483.648… +2.147.483.647). The unit is microsteps.

Closed loop: the range of the MVP command is 32 bit signed (−2.147.483.648… +2.147.483.647). The unit is encoder steps.

Positioning can be interrupted using MST, ROL or ROR commands.

THREE OPERATION TYPES ARE AVAILABLE:

- Moving to an absolute position in the range from −2.147.483.648… +2.147.483.647.
- 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 2.147.483.647 (2^{31}-1) microsteps resp. encoder steps. Otherwise the motor will run in the opposite direction in order to take the shorter distance.
When moving to a coordinate, the coordinate has to be set properly in advance with the help of the SCO, CCO or ACO command.

Internal function: A new position value is transferred to the axis parameter #0 (target position).

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

Mnemonic: MVP <ABS|REL|COORD>, 0, <position|offset|coordinate number>

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
4	0 ABS – absolute	<motor> 0*	<position> −2.147.483.648… +2.147.483.647
	1 REL – relative		<offset> −2.147.483.648… +2.147.483.647
	2 COORD – coordinate		<coordinate number> 0… 20

*** Motor number is always O as only one motor is involved**

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

Example:

Move motor 0 to (absolute) position 90000
Mnemonic: MVP ABS, 0, 9000

Binary:

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

Example:

Move motor 0 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
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$04	$01	$00	$ff	$ff	$fc	$18

Example:

Move motor 0 to previously stored coordinate #8
Mnemonic: MVP COORD, 0, 8

Binary:

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

5.7.5 SAP (set axis parameter)

With this command most of the motion control parameters 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. The parameter is transferred to the correct position in the appropriate device.

Related commands: GAP, STAP, RSAP, AAP

Mnemonic: SAP <parameter number>, 0, <value>

Binary representation:

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

*** Motor number is always O as only one motor is involved**

Reply in direct mode:

STATUS	VALUE
100 – OK	don’t care

For a table with parameters and values which can be used together with this command please refer to chapter 6.

Example:

Set the absolute maximum current of the motor during movements to approx. 78% of max. module current:

Mnemonic: SAP 6, 0, 200

Binary:

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

5.7.6 GAP (get axis parameter)

Most parameters of the TMCM-1311 can be adjusted individually for the axis. With this parameter they can be read out. In standalone mode the requested value is also transferred to the accumulator register for further processing purposes (such as conditioned jumps). In direct mode the value read is only output in the *value* field of the reply (without affecting the accumulator).

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

Related commands: SAP, STAP, AAP, RSAP

Mnemonic: GAP <parameter number>, 0

Binary representation:

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

*** Motor number is always O as only one motor is involved**

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

For a table with parameters and values which can be used together with this command please refer to chapter 6.

Example:

Get the maximum current of motor
Mnemonic: GAP 6, 0

Binary:

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

Reply:

Byte Index	0	1	2	3	4	5	6	7
Function	Host-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$02	$01	$64	$06	$00	$00	$02	$80

5.7.7 STAP (store axis parameter)

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

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

Related commands: SAP, RSAP, GAP, AAP

Mnemonic: STAP <parameter number>, 0

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
7	<parameter number>	<motor> 0*	don't care*[1]

*** Motor number is always O as only one motor is involved**
***[1] 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

For a table with parameters and values which can be used together with this command please refer to chapter 6.
The STAP command will not have any effect when the configuration EEPROM is locked (refer to 10.1). In direct mode, the error code 5 (configuration EEPROM locked, see also section 4.2.1) will be returned in this case.

Example:

Store the maximum speed of motor
Mnemonic: STAP 4, 0

Binary:

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

5.7.8 RSAP (restore axis parameter)

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

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

Relate commands: SAP, STAP, GAP, and AAP

Mnemonic: RSAP <parameter number>, 0

Binary representation:

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

*** Motor number is always O as only one motor is involved**

Reply structure in direct mode:

STATUS	VALUE
100 – OK	don't care

For a table with parameters and values which can be used together with this command please refer to chapter 6.

Example:

Restore the maximum current of motor
Mnemonic: RSAP 6, 0

Binary:

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

5.7.9 SGP (set global parameter)

With this command most of the module specific parameters not directly related to motion control can be specified and the TMCL user variables can be changed. Global parameters are related to the host interface, peripherals or application specific variables. The different groups of these parameters are organized in *banks* to allow a larger total number for future products. Currently, bank 0 and 1 are used for global parameters, bank 2 is used for user variables, and bank 3 is used for interrupts.

> 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.
> For a table with parameters and bank numbers which can be used together with this command please refer to chapter 10.

Internal function: the parameter format is converted. 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

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
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$09	$42	$00	$00	$00	$00	$03

5.7.10 GGP (get global parameter)

All global parameters can be read with this function. Global parameters are related to the host interface, peripherals or application specific variables. The different groups of these parameters are organized in *banks* to allow a larger total number for future products. Currently, bank 0 and 1 are used for global parameters, bank 2 is used for user variables, and bank 3 is used for interrupts.

For a table with parameters and bank numbers which can be used together with this command please refer to chapter 10

Internal function: the parameter is read out of the correct position in the appropriate device. The parameter format is converted.

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

Example:

Get the serial address of the target device
Mnemonic: GGP 66, 0

Binary:

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

Reply:

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

⇨ **Status = no error, value = 1**

5.7.11 STGP (store global parameter)

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

For a table with parameters and bank numbers which can be used together with this command please refer to chapter 10

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

Example:

Store the user variable #42
Mnemonic: STGP 42, 2

Binary:

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

5.7.12 RSGP (restore global parameter)

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

For a table with parameters and bank numbers which can be used together with this command please refer to chapter 10

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

Relate commands: SGP, STGP, GGP, and AGP

Mnemonic: RSGP <parameter number>, <bank number>

Binary representation:

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

Reply structure in direct mode:

STATUS	VALUE
100 – OK	don't care

Example:

Restore the user variable #42

Mnemonic: RSGP 42, 2

Binary:

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

5.7.13 RFS (reference search)

The TMCM-1311 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 (chapter 6). 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. Interaction is possible during execution.

Related commands: WAIT

Mnemonic: RFS <START|STOP|STATUS>, 0

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
13	0 START – start ref. search 1 STOP – abort ref. search 2 STATUS – get status	<motor> 0*	see below

*** Motor number is always O 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	ref. search active
	other values	no ref. search active

Example:

Start reference search of motor 0
Mnemonic: RFS START, 0

Binary:

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

With this module it is possible to use stall detection instead of a reference search.

5.7.14 SIO (set input / output)

SIO sets the status of the general digital output either to low (0) or to high (1). Bank 2 is used for this purpose.

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> 2	<value> 0/1

Reply structure:

STATUS	VALUE
100 – OK	don't care

Example:

Set OUT_7 to high (bank 2, output 7)
Mnemonic: SIO 7, 2, 1

Binary:

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$0e	$07	$02	$00	$00	$00	$01

OVERVIEW CONNECTORS 0 AND 1

Figure 5.1 I/O connectors

OVERVIEW OF CONNECTORS 0 AND 1

Pin	IN/OUT 0	IN/OUT 1	Direction	Description
1	GND	GND	Power (GND)	GND
2	VCC	VCC	Power (supply output)	Connected to $V_{DIGITAL}$ of Power connector
3	AIN0	AIN4	Input	Dedicated analog input, input voltage range: 0… +10V, resolution: 12bit (0… 4095)
4	IN1	IN5	Input	Digital input (+24V compatible)
5	IN2	IN6	Input	Digital input (+24V compatible)
6	IN3	IN7	Input	Digital input (+24V compatible)
7	OUT0	OUT4	Output	Open-drain output (max. 100mA) Integrated freewheeling diode
8	OUT1	OUT5	Output	Open-drain output (max. 100mA) Integrated freewheeling diode
9	OUT2	OUT6	Output	Open-drain output (max. 100mA) Integrated freewheeling diode
10	OUT3	OUT7	Output	Open-drain output (max. 1A) Integrated freewheeling diode

I/O PORTS USED FOR SIO AND COMMAND

I/O Connector	Pin	I/O port	Command	Range
0	7	OUT_0	SIO 0, 2, <n>	1/0
0	8	OUT_1	SIO 1, 2, <n>	1/0
0	9	OUT_2	SIO 2, 2, <n>	1/0
0	10	OUT_3	SIO 3, 2, <n>	1/0
1	7	OUT_4	SIO 4, 2, <n>	1/0
1	8	OUT_5	SIO 5, 2, <n>	1/0
1	9	OUT_6	SIO 6, 2, <n>	1/0
1	10	OUT_7	SIO 7, 2, <n>	1/0

ADDRESSING ALL OUTPUT LINES WITH ONE SIO COMMAND:

- Set the type parameter to 255 and the bank parameter to 2.
- The value parameter must then be set to a value between 0… 255, where every bit represents one output line.
- Furthermore, the value can also be set to -1. In this special case, the contents of the lower 8 bits of the accumulator are copied to the output pins.

Example:

Set all output pins high.
Mnemonic: SIO 255, 2, 3

THE FOLLOWING PROGRAM WILL SHOW THE STATES OF THE INPUT LINES ON THE OUTPUT LINES:

```
Loop: GIO 255, 0
      SIO 255, 2,-1
      JA Loop
```

5.7.15 GIO (get input /output)

With this command the status of the two available general purpose inputs of the module can be read out. The function reads a digital or analogue input port. Digital lines will read 0 and 1, while the ADC channels deliver their 12 bit result in the range of 0… 4095.

GIO IN STANDALONE MODE

In standalone mode the requested value is copied to the *accumulator* (accu) for further processing purposes such as conditioned jumps.

GIO IN DIRECT MODE

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
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$0f	$00	$01	$00	$00	$00	$00

Reply:

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

OVERVIEW CONNECTORS 0 AND 1

Figure 5.2 I/O connectors

OVERVIEW OF CONNECTORS 0 AND 1

Pin	IN/OUT 0	IN/OUT 1	Direction	Description
1	GND	GND	Power (GND)	GND
2	VCC	VCC	Power (supply output)	Connected to $V_{DIGITAL}$ of Power connector
3	AIN0	AIN4	Input	Dedicated analog input, input voltage range: 0… +10V, resolution: 12bit (0… 4095)
4	IN1	IN5	Input	Digital input (+24V compatible)
5	IN2	IN6	Input	Digital input (+24V compatible)
6	IN3	IN7	Input	Digital input (+24V compatible)
7	OUT0	OUT4	Output	Open-drain output (max. 100mA) Integrated freewheeling diode
8	OUT1	OUT5	Output	Open-drain output (max. 100mA) Integrated freewheeling diode
9	OUT2	OUT6	Output	Open-drain output (max. 100mA) Integrated freewheeling diode
10	OUT3	OUT7	Output	Open-drain output (max. 1A) Integrated freewheeling diode

5.7.15.1 I/O Bank 0 – Digital Inputs:

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

Note: AIN_0 and AIN_4 can be used in digital mode, too.

I/O Connector	Pin	I/O port	Command	Range
0	3	AIN_0	GIO 0, 0	0/1
0	4	IN_1	GIO 1, 0	0/1
0	5	IN_2	GIO 2, 0	0/1
0	6	IN_3	GIO 3, 0	0/1
1	3	AIN_4	GIO 4, 0	0/1
1	4	IN_5	GIO 5, 0	0/1
1	5	IN_6	GIO 6, 0	0/1
1	6	IN_7	GIO 7, 0	0/1

READING ALL DIGITAL INPUTS WITH ONE GIO COMMAND:

- Set the type parameter to 255 and the bank parameter to 0.
- In this case the status of all digital input lines will be read to the lower eight bits of the accumulator.

USE FOLLOWING PROGRAM TO REPRESENT THE STATES OF THE INPUT LINES ON THE OUTPUT LINES:

```
Loop: GIO 255, 0
      SIO 255, 2,-1
      JA Loop
```

5.7.15.2 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.

I/O Connector	Pin	I/O port	Command	Range
0	3	AIN_0	GIO 0, 1	0… 4095
1	3	AIN_4	GIO 4, 1	0… 4095

FURTHER READ-OUT COMMANDS

I/O port	Command
Supply voltage [1/10V]	GIO 8, 1
Temperature [˚C]	GIO 9, 1
Coil current A [mA]	GIO 10, 1
Coil current B [mA]	GIO 11, 1
Input current [mA]	GIO 12, 1

5.7.15.3 I/O Bank 2 –States of Digital Outputs

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

I/O Connector	Pin	I/O port	Command	Range
0	7	OUT_0	GIO 0, 2, <n>	1/0
0	8	OUT_1	GIO 1, 2, <n>	1/0
0	9	OUT_2	GIO 2, 2, <n>	1/0
0	10	OUT_3	GIO 3, 2, <n>	1/0
1	7	OUT_4	GIO 4, 2, <n>	1/0
1	8	OUT_5	GIO 5, 2, <n>	1/0
1	9	OUT_6	GIO 6, 2, <n>	1/0
1	10	OUT_7	GIO 7, 2, <n>	1/0

5.7.16 CALC (calculate)

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

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

Mnemonic: CALC <operation>, <value>

Binary representation:

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

Example:

Multiply accu by -5000
Mnemonic: CALC MUL, -5000

Binary:

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

Reply:

Byte Index	0	1	2	3	4	5	6	7
Function	Host-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$02	$01	$64	$13	$ff	$ff	$ec	$78

Status = no error, value = -5000

5.7.17 COMP (compare)

The specified number is compared to the value in the accumulator register. The result of the comparison can for example be used by the conditional jump (JC) instruction.

This command is intended for use in standalone operation only.

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

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

Mnemonic: COMP <value>

Binary representation:

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

Example:

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

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

Binary format of the COMP 1000 command:

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

5.7.18 JC (jump conditional)

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

This function is for standalone operation only.

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>

Binary representation:

INSTRUCTION NO.	TYPE <condition>	MOT/BANK	VALUE
21	0 ZE - zero 1 NZ - not zero 2 EQ - equal 3 NE - not equal 4 GT - greater 5 GE - greater/equal 6 LT - lower 7 LE - lower/equal 8 ETO - time out error	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
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$15	$05	$00	$00	$00	$00	$0a

5.7.19 JA (jump always)

Jump to a fixed address in the TMCL program memory.

This command is intended only for standalone operation.

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
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$16	$00	$00	$00	$00	$00	$14

5.7.20 CSUB (call subroutine)

This function calls a subroutine in the TMCL program memory.

This command is intended for standalone operation, only.

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
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$17	$00	$00	$00	$00	$00	$64

5.7.21 RSUB (return from subroutine)

Return from a subroutine to the command after the CSUB command.

This command is intended for standalone operation only.

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: 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 RSUB:

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

5.7.22 WAIT (wait for an event to occur)

This instruction interrupts the execution of the TMCL program until the specified condition is met.

This command is intended for standalone operation. only.

THERE ARE FIVE DIFFERENT WAIT CONDITIONS THAT CAN BE USED:

TICKS Wait until the number of timer ticks specified by the <ticks> parameter has been reached.

POS Wait until the target position of the motor specified by the <motor> parameter has been reached. An optional timeout value (0 for no timeout) must be specified by the <ticks> parameter.

REFSW Wait until the reference switch of the motor specified by the <motor> parameter has been triggered. An optional timeout value (0 for no timeout) must be specified by the <ticks> parameter.

LIMSW Wait until a limit switch of the motor specified by the <motor> parameter has been triggered. An optional timeout value (0 for no timeout) must be specified by the <ticks> parameter.

RFS Wait until the reference search of the motor specified by the <motor> field has been reached. An optional timeout value (0 for no timeout) must be specified by the <ticks> parameter.

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

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

Related commands: JC, CLE

Mnemonic: WAIT <condition>, 0, <ticks>

Binary representation:

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

*** Motor number is always O as only one motor is involved.**
***[1] One tick is 10 milliseconds.**

Example:

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

Binary:

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

5.7.23 STOP (stop TMCL program execution)

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

The STOP command should be placed at the end of every standalone TMCL program.
The STOP command 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
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$1c	$00	$00	$00	$00	$00	$00

5.7.24 SCO (set coordinate)

Up to 20 position values (coordinates) can be stored for every axis for use with the MVP COORD command. This command sets a coordinate to a specified value. Depending on the global parameter 84, the coordinates are only stored in RAM or also stored in the EEPROM and copied back on startup (with the default setting the coordinates are stored in RAM only).

Please note that the coordinate number 0 is always stored in RAM only.

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

Related commands: GCO, CCO, MVP

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

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
30	<coordinate number> 0… 20	<motor> 0*	<position> −2.147.483.648… +2.147.483.647

*** Motor number is always O 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
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$1e	$01	$00	$00	$00	$03	$e8

Note:
Two special functions of this command have been introduced that make it possible to copy all coordinates or one selected coordinate to the EEPROM.

THESE FUNCTIONS CAN BE ACCESSED USING THE FOLLOWING SPECIAL FORMS OF THE SCO COMMAND:

SCO 0, 255, 0	copies all coordinates (except coordinate number 0) from RAM to the EEPROM.
SCO <coordinate number>, 255, 0	copies the coordinate selected by <coordinate number> to the EEPROM. The coordinate number must be a value between 1 and 20.

5.7.25 GCO (get coordinate)

This command makes possible to read out a previously stored coordinate. In standalone 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. Depending on the global parameter 84, the coordinates are only stored in RAM or also stored in the EEPROM and copied back on startup (with the default setting the coordinates are stored in RAM, only).

Please note that the coordinate number 0 is always stored in RAM, only.

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

Related commands: SCO, CCO, MVP

Mnemonic: GCO <coordinate number>, 0

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
31	<coordinate number> 0… 20	<motor> 0*	don't care

*** Motor number is always O as only one motor is involved**

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

Example:

Get motor value of coordinate 1
Mnemonic: GCO 1, 0

Binary:

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

Reply:

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

⇨ **Value: 0**

Two special functions of this command have been introduced that make it possible to copy all coordinates or one selected coordinate from the EEPROM to the RAM.

These functions can be accessed using the following special forms of the GCO command:

GCO 0, 255, 0	copies all coordinates (except coordinate number 0) from the EEPROM to the RAM.
GCO <coordinate number>, 255, 0	copies the coordinate selected by <coordinate number> from the EEPROM to the RAM. The coordinate number must be a value between 1 and 20.

5.7.26 CCO (capture coordinate)

The actual position of the axis is copied to the selected coordinate variable. Depending on the global parameter 84, the coordinates are only stored in RAM or also stored in the EEPROM and copied back on startup (with the default setting the coordinates are stored in RAM only). Please see the SCO and GCO commands on how to copy coordinates between RAM and EEPROM.

Note, that the coordinate number 0 is always stored in RAM only.

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>

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
32	<coordinate number> 0… 20	<motor> 0*	don't care

*** Motor number is always O as only one motor is involved**

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

Example:

Store current position of the axis 0 to coordinate 3
Mnemonic: CCO 3, 0

Binary:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$20	$03	$00	$00	$00	$00	$00

5.7.27 ACO (accu to coordinate)

With the ACO command the actual value of the accumulator is copied to a selected coordinate of the motor. Depending on the global parameter 84, the coordinates are only stored in RAM or also stored in the EEPROM and copied back on startup (with the default setting the coordinates are stored in RAM only).

Please note also that the coordinate number 0 is always stored in RAM only. For Information about storing coordinates refer to the SCO command.

Internal function: the actual value of the accumulator is stored in the internal position array.

Related commands: GCO, CCO, MVP COORD, SCO

Mnemonic: ACO <coordinate number>, 0

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
39	<coordinate number> 0… 20	<motor> 0*	don't care

*** Motor number is always O as only one motor is involved**

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

Example:

Copy the actual value of the accumulator to coordinate 1 of motor
Mnemonic: ACO 1, 0

Binary:

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$27	$01	$00	$00	$00	$00	$00

5.7.28 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>

Binary representation:

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

Example:

Multiply accu by X-register

Mnemonic: CALCX MUL

Binary:

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

5.7.29 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.

For a table with parameters and values which can be used together with this command please refer to chapter 6.

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

Mnemonic: AAP <parameter number>, 0

Binary representation:

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

*** Motor number is always O as only one motor is involved**

Reply in direct mode:

STATUS	VALUE
100 – OK	don’t care

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**
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$22	$00	$00	$00	$00	$00	$00

5.7.30 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 standalone application.
For a table with parameters and bank numbers which can be used together with this command please refer to chapter 10.

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

Example:

Copy accumulator to TMCL user variable #3
Mnemonic: AGP 3, 2

Binary:

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

5.7.31 CLE (clear error flags)

This command clears the internal error flags.

The CLE command is intended for use in standalone mode, only.

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.
- EDV: clear the deviation flag

Related commands: JC

Mnemonic: CLE <flags>
where <flags>=ALL|ETO|EDV|EPO

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
36	0 – (ALL) all flags 1 – (ETO) timeout flag 3 – (EDV) deviation 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
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$24	$01	$00	$00	$00	$00	$00

5.7.32 VECT (set interrupt vector)

The VECT command defines an interrupt vector. It needs an interrupt number and a label as parameter (like in JA, JC and CSUB commands).

This label must be the entry point of the interrupt handling routine.

Related commands: EI, DI, RETI

Mnemonic: VECT <interrupt number>, <label>

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
37	<interrupt number>	don't care	<label>

The following table shows all interrupt vectors that can be used.

Interrupt number	Interrupt type
0	Timer 0
1	Timer 1
2	Timer 2
3	Target position reached
15	stallGuard2
21	Deviation
27	Left stop switch
28	Right stop switch
39	Input change 0
40	Input change 1
41	Input change 2
42	Input change 3
43	Input change 4
44	Input change 5
45	Input change 6
46	Input change 7
255	Global interrupts

Example:

Define interrupt vector at target position 500
VECT 3, 500

Binary format of VECT:

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$25	$03	$00	$00	$00	$01	$F4

5.7.33 EI (enable interrupt)

The EI command enables an interrupt. It needs the interrupt number as parameter. Interrupt number 255 globally enables interrupts.

Related command: DI, VECT, RETI

Mnemonic: EI <interrupt number>

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
25	<interrupt number>	don't care	don't care

The following table shows all interrupt vectors that can be used:

Interrupt number	Interrupt type
0	Timer 0
1	Timer 1
2	Timer 2
3	Target position reached
15	stallGuard2
21	Deviation
27	Left stop switch
28	Right stop switch
39	Input change 0
40	Input change 1
41	Input change 2
42	Input change 3
43	Input change 4
44	Input change 5
45	Input change 6
46	Input change 7
255	Global interrupts

Examples:

Enable interrupts globally
EI, 255

Binary format of EI:

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$19	$FF	$00	$00	$00	$00	$00

Enable interrupt when target position reached
EI, 3

Binary format of EI:

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$19	$03	$00	$00	$00	$00	$00

5.7.34 DI (disable interrupt)

The DI command disables an interrupt. It needs the interrupt number as parameter. Interrupt number 255 globally disables interrupts.

Related command: EI, VECT, RETI

Mnemonic: DI <interrupt number>

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
26	<interrupt number>	don't care	don't care

The following table shows all interrupt vectors that can be used:

Interrupt number	Interrupt type
0	Timer 0
1	Timer 1
2	Timer 2
3	Target position reached
15	stallGuard2
21	Deviation
27	Left stop switch
28	Right stop switch
39	Input change 0
40	Input change 1
41	Input change 2
42	Input change 3
43	Input change 4
44	Input change 5
45	Input change 6
46	Input change 7
255	Global interrupts

Examples:

Disable interrupts globally
DI, 255

Binary format of DI:

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

Disable interrupt when target position reached
DI, 3

Binary format of DI:

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

5.7.35 RETI (return from interrupt)

This command terminates the interrupt handling routine, and the normal program execution continues.

At the end of an interrupt handling routine the RETI command must be executed.

Internal function: the saved registers (A register, X register, flags) are copied back. Normal program execution continues.

Related commands: EI, DI, VECT

Mnemonic: RETI

Binary representation:

INSTRUCTION NO.	TYPE	MOT/BANK	VALUE
38	don't care	don't care	don't care

Example: Terminate interrupt handling and continue with normal program execution
RETI

Binary format of RETI:

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$26	$00	$00	$00	$00	$01	$00

5.7.36 Customer Specific TMCL Command Extension (user function)

The user definable functions UF0… UF7 are predefined functions without topic for user specific purposes. A user function (UF) command uses three parameters. Please contact TRINAMIC for a customer specific programming.

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

Related commands: none

Mnemonic: UF0… UF7 <parameter number>

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
Function	Target-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$02	$01	user defined	64… 71	user defined	user defined	user defined	user defined

5.7.37 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 standalone 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	0*

*** Motor number**

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

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

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

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

5.7.38 BIN (return to binary mode)

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

Related Commands: none

Mnemonic: BIN

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

5.7.39 Control Functions: Firmware Version and Reset to Factory Defaults

There are several TMCL control functions, but for the user are only 136 and 137 interesting. Other control functions can be used with axis parameters.

Instruction number	Type	Command	Description
136	0 – string 1 – binary	Firmware version	Get the module type and firmware revision as a string or in binary format. (*Motor/Bank* and *Value* are ignored.)
137	don't care	Reset to factory defaults	Reset all settings stored in the EEPROM to their factory defaults This command does not send back a reply. *Value must be 1234*

FURTHER INFORMATION ABOUT COMMAND 136

- **Type set to 0 - reply as a string:**

Byte index	Contents
1	Host Address
2… 9	Version string (8 characters, e.g. 1311V107)

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	05
2	1F
3	Version number, low byte
4	Version number, high byte

6 Axis Parameters

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

Attention!

Related to positioning, there are several parameters which have two unit specifications.
- In *open loop mode*, calculate with *µsteps*.
- In closed loop mode, calculate with encoder steps.

For further information about closed loop operation parameters and functions please refer to Section 8.

MEANING OF THE LETTERS IN COLUMN *ACCESS*:

Access type	Related command(s)	Description
R	GAP	Parameter readable
W	SAP, AAP	Parameter writable
E	STAP, RSAP	Parameter automatically restored from EEPROM after reset or power-on. These parameters can be stored permanently in EEPROM using STAP command and also explicitly restored (copied back from EEPROM into RAM) using RSAP.

Basic parameters should be adjusted to motor / application for proper module operation.

Parameters for the more experienced user – please do not change unless you are absolutely sure!

Parameters for closed loop operation

Number	Axis Parameter	Description	Range [Unit]	Acc.
0	Target position	The desired position in position mode (see ramp mode, no. 138).	−2.147.483.648… +2.147.483.647 [µsteps] [encoder steps]	RW
1	Actual position	The current position of the motor. Should only be overwritten for reference point setting.	−2.147.483.648… +2.147.483.647 [µsteps] [encoder steps]	RW
2	Target speed	The desired speed in velocity mode (see ramp mode / axis parameter 128). In position mode, this parameter is set by hardware: - during acceleration to maximum speed - during deceleration and rest to zero	-327.678.000… +327.679.999 [pps]	RW
3	Actual speed	The current rotation speed.	-327.678.000… +327.679.999 [pps]	RW
4	Maximum positioning speed	Should not exceed the physically highest possible value.	0… +327.679.999 [pps]	RWE
5	Maximum acceleration	The limit for acceleration (and deceleration).	1… +24.999.998 [pps/s]	RWE

Number	Axis Parameter	Description	Range [Unit]	Acc.
6	Absolute max. current (CS / Current Scale)	The maximum value is 255. This value means 100% of the maximum current of the module. The current adjustment is within the range 0… 255 and can be adjusted in 32 steps. 0… 7, 79…87, 160… 167, 240… 247 8… 15, 88… 95, 168… 175, 248… 255 16… 23, 96… 103, 176… 183 24… 31, 104… 111, 184… 191 32… 39, 112… 119, 192… 199 40… 47, 120… 127, 200… 207 48… 55, 128… 135, 208… 215 56… 63, 136… 143, 216… 223 64… 71, 144… 151, 224… 231 72… 79, 152… 159, 232… 239 *The most important motor setting, since too high values might cause motor damage!*	0… 255 $I_{peak} =< value > \times \frac{4.2A}{255}$ $I_{RMS} =< value > \times \frac{3A}{255}$	RWE
7	Standby current	The current limit two seconds after the motor has stopped.	0… 255 $I_{peak} =< value > \times \frac{4.2A}{255}$ $I_{RMS} =< value > \times \frac{3A}{255}$	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 (home) switch. Connect this switch to IN1.	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.	0/1	R
12	Right limit switch disable	Deactivates the stop function of the right switch if set.	0/1	RWE
13	Left limit switch disable	Deactivates the stop function of the left switch if set.	0/1	RWE
14	CL torque mode target current	Target RMS current value for torque mode. Positive and negative values define rotation direction. - Writing a target value to this parameter automatically switches to torque mode. - Reading provides the actual configured target current while in torque mode. - Reading while in other modes (velocity mode, position mode) provides information on the *actual advance angle* (=delta). In these cases the unit is microsteps. The maximum current that can be configured can be read out using axis parameter 15.	-3000… +3000 [mA]	RW *torque mode* R *velocity and position mode*
15	Maximum possible current	Based on axis parameters 6 and 179 this parameter returns the maximum possible RMS current.	0….+3000 [mA]	R
16	CL velocity reached	This flag is set when the actual velocity is within the velocity reached window (axis parameter 17) around the target position.	0/1	R
17	CL Velocity reached window	Window around the target velocity value in which the target velocity will be considered as being reached. The *velocity_reached* flag will be set accordingly.	0… +268.435.454 [pps]	RW

Number	Axis Parameter	Description	Range [Unit]	Acc.
18	Status word	The status word contains 12 bits (bit 0… 11). They can be read out using command GAP 18. If a bit is set the specific event has occurred. Bit 0 – Target reached Bit 1 – Velocity reached Bit 2 – Closed loop mode Bit 3 – Position mode Bit 4 – Velocity mode Bit 5 – Torque mode Bit 6 – Home switch Bit 7 – Left stop switch Bit 8 – Right stop switch Bit 9 – Undervoltage Bit 10 – Overvoltage Bit 11 – Overtemperature Bit 12 – Position hold mode active Bit 13 – I^2t value exceeded	0/1	R
19	CL torque mode actual current	Actual current in torque mode	-3000… +3000 [mA]	R
20	CL torque mode slope	Slope in torque mode (related to acceleration and deceleration).	[mA/s]	RW
25	Thermal winding time constant	Thermal winding time constant for the used motor. Used for I^2t monitoring.	0… +4294967295 [ms]	RWE
26	I^2t limit	An actual I^2t sum that exceeds this limit leads to increasing the I^2t exceed counter.	0… +4294967295	RWE
27	I^2t sum	Actual sum of the I^2t monitor.	0… +4294967295	R
28	I^2t exceed counter	Counts how often an I^2t sum was higher than the I^2t limit.	0… +4294967295	RWE
29	Clear I^2t exceeded flag	Clear the flag that indicates that the I^2t sum has exceeded the I^2t limit.	(ignored)	W
108	CL field weakening minimum velocity	Minimum motor speed at which the speed dependent Back EMF compensation will be applied (field weakening). Based on the velocity measured via encoder feedback.	0… +327.679.999 [pps]	RW
109	CL field weakening maximum velocity	Maximum motor speed for the speed dependent Back EMF compensation will be applied (field weakening). Based on the velocity measured via encoder feedback.	0… +327.679.999 [pps]	RW
112	CL encoder offset	Offset between encoder and electrical angle for correction of possible misalignment.	[encoder steps] Default = 0	RWE
113	CL current scale minimum	Minimum current scale factor for current regulation. 255 = 1 = 100% of maximum current 127 = 0.5 = 50% of maximum current … *Attention!* The maximum current itself is defined by the CS parameter of the motor driver chip (see axis parameter 6)	Default = 15 [1/256]	RW

Number	Axis Parameter	Description	Range [Unit]	Acc.
114	CL current scale maximum	Maximum current scale factor for current regulation. 255 = 1 = 100% of maximum current 127 = 0.5 = 50% of maximum current … *Attention!* - The maximum current itself is defined by the CS parameter of motor driver chip (see axis parameters 6) - The physical maximum current is defined by the sense resistors and MOSFETs of the TMCM-1311 and is limited to 3.0A RMS.	Default = 255 [1/256]	RW
115	CL current scale input select	Current scaling using raw position error or product of position error and gain factor (depends on axis parameters 136 and 137). 0 = only raw position error used for current scaling 1 = (position error * gain) used for current scaling	0/1 Default = 1	RW
116	CL current scale lower error limit	Position error from which on the current amplitude is increased (current scale factor is increased).	Default = 0 [encoder steps]	RW
117	CL current scale upper error limit	Position error from which on the current amplitude will be increased to its configured maximum (parameter 114). This parameter must be higher than axis parameter 116.	Default = 255 [encoder steps]	RW
118	CL current scale increment value	Current scale increment value if the actual current scale factor is below the calculated current scale factor target value. This parameter defines the step width at which the current scale factor will be increased.	Default = 1 [1/256]	RW
119	CL current scale decrement value	Current scale decrement value if the actual current scale factor is higher than the calculated current scale target value. This parameter defines the step width at which the current scale factor will be decreased.	Default = 1 [1/256]	RW
120	CL current scale increment timeout	This parameter defines the delay between two current scale factor increments and thereby controls the rate at which the current scale factor will be increased. Setting a timeout value here serves for dampening and prevents from high oscillations. 0 = the scale factor will directly be set to the actual target value without delay.	Default = 1 [ms]	RW
121	CL current scale decrement timeout	This parameter defines the delay between two current scale factor decrements and thereby controls the rate at which the current scale factor will be decreased, e.g., in order to prevent from oscillations around the target position. Setting a timeout value here serves for dampening and prevents from high oscillations. 0 = the scale factor will directly be set to the actual target value without delay.	Default = 1 [ms]	RW

Number	Axis Parameter	Description	Range [Unit]	Acc.
122	CL current scale enable	1 = current scaling function on for closed loop 0 = current scaling function off, closed loop operation is still possible The current scaling functionality can be switched off if full specified current amplitude shall be used all the time. When switched on, the current scaling functionality adapts the current according to the configured profile. This saves energy and keeps the motor cooler.	Default = 1	RW
123	CL actual current scale factor	Actual value of the current scale factor.	0… 255 [1/256]	R
124	CL correction velocity proportional factor	Proportional factor for on-line / live position lag compensation in positioning mode during a ramp movement. For a very quick compensation while the drive is active choose a high / the maximum value.	0… 65535	RW
125	CL max. following error	Maximum allowed following error during a ramp movement in positioning mode before starting compensation the position lag using parameters 124 and 126.	0… +268.435.454 Default = 0 [µsteps]	RW
126	CL max. correction velocity	Maximum correction speed during positioning mode. If a certain ramp/motion profile is used and a lag occurs during movement, the velocity will be increased to the maximum correction speed to compensate the position lag/ following error. If set to 0 or smaller than target velocity, the ramp profile will be simply extended if there is a lag between actual position and commanded position. If greater than target velocity the position lag will be compensated on-line / live.	0… +327.679.999 [pps]	RW
127	Relative positioning option	Positioning relative to one out of three starting points can be initialized using this parameter. 0 - last target position 1 - actual ramp generator position 2 - actual encoder position	0/1/2	RW
128	Ramp mode	Automatically set when using ROR, ROL, MVP or SAP 14 commands. 0: Position mode. Steps are generated, when the parameters actual position and target position differ. Trapezoidal speed ramps are provided. When switching into this mode (using MVP ABS|REL|COORD), the motor will immediately start when there is a position difference. 1: Velocity mode. The motor will run continuously and the speed will be changed with constant (maximum) acceleration, if the parameter target speed is changed. 2: Torque mode. In closed loop mode only, the motor can be driven with constant torque. The velocity is only limited by the motor characteristics and the current setting. This mode of operation cannot be used in open loop mode!	0/1/2	RW

Number	Axis Parameter	Description	Range [Unit]	Acc.
129	Closed loop operation	0: open loop mode 1: closed loop mode When switching from open loop to closed loop mode the closed loop system will be initialized. *Attention:* Wait for the closed loop init flag to be set to 1 after switching to closed loop mode (see parameter 133).	0/1	RW
130	Start/Stop velocity	Ramp generation for acceleration and deceleration begins/ends with this start and stop velocity value. When set to equal to the target speed, no ramp is generated. When set smaller than the target velocity (also to zero), the ramp starts with this velocity value. Must be smaller than target velocity (axis parameter 4).	0… +327.679.999 Default = 0 [pps]	RWE
131	Measured speed	The speed measured using the encoder (read only).	+/-INT_MAX [pps/s]	R
132	Encoder initialization	Use this parameter to initialize the encoder automatically by writing 1. Attention: the fullstep resolution of the encoder has to be set first! Further, the status of encoder initialization can be read out: 0 | no encoder initialization 1 | encoder initialization in process 2 | encoder initialization completed 3 | encoder initialization incorrect	Write: 0/1 Read-out: 0/1/2/3	RW
133	CL init flag	0: open loop mode or closed loop not initialized 1: closed loop is initialized and ready for use See also parameter 129.	0/1	R
134	CL position reached window	Window around the target position in which the target position will be considered as being reached. This value should be adjusted to the application and the resolution of the encoder. If the actual position is within the target reached window for at least or longer than defined by axis parameter 135, the *position_reached* flag will be set.	0… 255 [encoder steps] Default = 50	RW
135	CL position reached time	Minimum duration the motor must be within the target reached window (axis parameter 134) before the position will be considered as reached and *position_reached* flag will be set.	0… 131072 Default = 100 [1/10 ms]	RW
136	CL standstill position error gain	When the target position is reached, the velocity regulation will be switched off and the system is in a special standstill mode. In this mode, the position is hold and maintained as long as the position error is within the range of the standstill error limit (axis parameter 138) around the target position. This parameter is a gain factor for the position error used for position maintenance in standstill mode. - A value of 20 typically provides good results. - Higher values provide higher stiffness. - For values greater than 25 a dampening factor > 0 (axis parameter 137) should be used as well to prevent from oscillations.	10… 50 Default = 10	RW

Number	Axis Parameter	Description	Range [Unit]	Acc.
137	CL standstill position error dampening factor	When the target position is reached, the velocity regulation is switched off and the system is in a standstill mode where the position is hold and maintained as long as the position error is within the range of the standstill error limit (axis parameter 138) around the target position. Parameter 137 is a dampening factor to prevent from oscillations when using a high proportional gain (axis parameter 136). - When axis parameter 136 is between 10 and 20, this parameter can be 0. - For higher proportional gain, e.g., 30, a dampening factor of 20 is a good starting value.	0… 65535 Default = 0	RW
138	CL standstill position error limit	When the target position is reached, the position maintenance by the ramp generator is switched off and the system is in a standstill mode. As long as the position error is within the range around the target position defined by axis parameter 138, the position maintenance is done using parameter 136 and 137. When the position error exceeds the range defined by this parameter, the configured ramp parameters are used to move back to the target position (as in normal positioning mode).	0… +268.435.454 [µsteps] Default=255	RW
140	Microstep resolution	0 full step 1 half step 2 4 microsteps 3 8 microsteps 4 16 microsteps 5 32 microsteps 6 64 microsteps 7 128 microsteps 8 256 microsteps (default) *This parameter is only for open loop mode. With closed loop, the microstep resolution is always 256.*	0… 8	RWE
141	CL torque mode start/stop current	Start and stop current in torque mode.	0… +3000 [mA]	RW
151	Current Phase A	Actual peak current Phase A	[mA]	R
152	Current Phase B	Actual peak current Phase B	[mA]	R
153	Supply Voltage	Actual value of supply voltage	[1/10 V]	R
154	DC Current	Actual DC current of the complete module + motor	[mA]	R
155	Module temperature	Actual temperature of the module	[°C]	R
162	Chopper blank time	Selects the comparator *blank time*. This time needs to safely cover the switching event and the duration of the ringing on the sense resistor. For low current drivers, a setting of 1 or 2 is good. For higher current applications like the TMCM-1311 a setting of 2 or 3 will be required.	0… 3	RW
163	Chopper mode	Selection of the chopper mode: 0 – spread cycle 1 – classic const. off time	0/1	RW

Number	Axis Parameter	Description	Range [Unit]	Acc.
164	Chopper hysteresis decrement	Hysteresis decrement setting. This setting determines the slope of the hysteresis during on time and during fast decay time. 0 – fast decrement 3 – very slow decrement	0… 3	RW
165	Chopper hysteresis end	Hysteresis end setting. Sets the hysteresis end value after a number of decrements. Decrement interval time is controlled by axis parameter 164. -3… -1: negative hysteresis end setting 0: zero hysteresis end setting 1… 12: positive hysteresis end setting	-3… 12	RW
166	Chopper hysteresis start	Hysteresis start setting. Please remark, that this value is an offset to the hysteresis end value.	0… 8	RW
167	Chopper off time	The off time setting controls the minimum chopper frequency. An off time within the range of 5µs to 20µs will fit. Off time setting for constant t_{Off} chopper: N_{CLK}= 12 + 32*t_{OFF} (Minimum is 64 clocks) Setting this parameter to zero completely disables all driver transistors and the motor can free-wheel.	0 / 2… 15	RW
168	smartEnergy current minimum (SEIMIN)	Sets the lower motor current limit for coolStep operation by scaling the CS (Current Scale, see axis parameter 6) value. minimum motor current: 0 – 1/2 of CS 1 – 1/4 of CS	0/1	RW
169	smartEnergy current down step	Sets the number of stallGuard2 readings above the upper threshold necessary for each current decrement of the motor current. Number of stallGuard2 measurements per decrement: Scaling: 0… 3: 32, 8, 2, 1 0: slow decrement 3: fast decrement	0… 3	RW
170	smartEnergy hysteresis	Sets the distance between the lower and the upper threshold for stallGuard2 reading. Above the upper threshold the motor current becomes decreased. Hysteresis: (smartEnergy hysteresis value + 1) * 32 Upper stallGuard2 threshold: (smartEnergy hysteresis start + smartEnergy hysteresis + 1) * 32	0… 15	RW
171	smartEnergy current up step	Sets the current increment step. The current becomes incremented for each measured stallGuard2 value below the lower threshold (see smartEnergy hysteresis start). current increment step size: Scaling: 0… 3: 1, 2, 4, 8 0: slow increment 3: fast increment / fast reaction to rising load	1… 3	RW

Number	Axis Parameter	Description	Range [Unit]	Acc.
172	smartEnergy hysteresis start	The lower threshold for the stallGuard2™ value (see smart Energy current up step).	0… 15	RW
173	stallGuard2 filter enable	Enables the stallGuard2 filter for more precision of the measurement. If set, reduces the measurement frequency to one measurement per four fullsteps. *In most cases it is expedient to set the filtered mode before using coolStep.* *Use the standard mode for step loss detection.* 0 – standard mode 1 – filtered mode	0/1	RW
174	stallGuard2 threshold	This signed value controls stallGuard2 *threshold* level for stall output and sets the optimum measurement range for readout. A lower value gives a higher sensitivity. Zero is the starting value. A higher value makes stallGuard2 less sensitive and requires more torque to indicate a stall. 0 – Indifferent value 1… 63 – less sensitivity -1 -64 – higher sensitivity	-64… 63	RW
175	Slope control high side	Determines the slope of the motor driver outputs. *Set to 2 or 3 for this module or rather use the default value.* 0: lowest slope 3: fastest slope	0… 3	RW
176	Slope control low side	Determines the slope of the motor driver outputs. *Set identical to slope control high side.*	0… 3	RW
177	Short protection disable	0: Short to GND protection is on 1: Short to GND protection is disabled *Use default value!*	0/1	RW
178	Short detection timer	0: 3.2µs 1: 1.6µs 2: 1.2µs 3: 0.8µs *Use default value!*	0… 3	RW
179	Vsense	Sense resistor voltage based current scaling 0: 0… 3A 1: 0… 1.5A *higher solution*	0/1	RW
180	smartEnergy actual current	This status value provides the *actual motor current* setting as controlled by coolStep. The value goes up to the CS value and down to the portion of CS as specified by SEIMIN. actual motor current scaling factor: 0 … 31: 1/32, 2/32, … 32/32	0… 31	RW
181	Stop on stall	Below this speed motor will not be stopped. Above this speed motor will stop in case stallGuard2 load value reaches zero.	0… +327.679.999 [pps]	RW
182	smartEnergy threshold speed	Above this speed coolStep becomes enabled.	0… +327.679.999 [pps]	RW
183	smartEnergy slow run current	Sets the motor current which is used below the threshold speed.	0… 255 $I_{peak} =< value > \times \frac{4.2A}{255}$ $I_{RMS} =< value > \times \frac{3A}{255}$	RW

Number	Axis Parameter	Description	Range [Unit]	Acc.
193	Ref. search mode	1 – search left stop switch only 2 – search rightstop switch, then search left stop switch 3 – search right stop switch, then search left stop switch from both sides 4 – search left stop switch from both sides 5 – search home switch in negative direction, reverse the direction when left stop switch reached 6 – search home switch in positive direction, reverse the direction when right stop switch reached 7 – search home switch in positive direction, ignore end switches 8 – search home switch in negative direction, ignore end switches *Adding 128 to these values reverses the polarity of the home switch input.*	1… 8	RWE
194	Referencing search speed	For the reference search this value directly specifies the search speed.	327.678.000… +327.679.999 [pps]	RWE
195	Referencing switch speed	Similar to parameter no. 194, the speed for the switching point calibration can be selected.	327.678.000… +327.679.999 [pps]	RWE
196	End switch distance	This parameter provides the distance between the end switches after executing the RFS command (mode 2 or 3).	0… +2.147.483.647 [µsteps] [encoder steps]	R
198	Clear on null	This parameter can be used to set bit 2 of axis parameter 201. 0 – Delete bit 2 of axis parameter 201. 1 – Clear encoder on next null channel event (set bit 2 of axis parameter 201).	0/1	RW
200	Boost current	Current used for acceleration and deceleration phases. If set to 0 the same current as set by axis parameter 6 will be used.	0… 255 $I_{peak} =< value > \times \frac{4.2A}{255}$ $I_{RMS} =< value > \times \frac{3A}{255}$	RWE
201	Encoder mode	Operation mode of the encoder. 4 (bit 2) – Clear encoder on next null channel event. 8 (bit 3) – Clear encoder on every null channel event. 16 (bit 4) – Null channel polarity (active high when set)	4, 8, 16	RWE
202	Motor Resolution	Motor fullstep resolution.	0… 400 Default: 200 [fullsteps]	RW
204	Freewheeling	Time after which the power to the motor will be cut when its velocity has reached zero. Parameter is not valid in closed loop mode!	0… 65535 0 = never [msec]	RWE
206	Actual load value	Readout of the actual load value used for stall detection (stallGuard2).	0… 1023	R
207	Max encoder deviation error flag	When maximum deviation is reached, motor is stopped/switched off. This flag shows this condition.	0/1	R

Number	Axis Parameter	Description	Range [Unit]	Acc.
208	Driver error flags	Bit 0: stallGuard2 status (1:threshold reached) Bit 1: Overtemperature (1: driver is shut down due to overtemperature) Bit 2: Pre-warning overtemperature (1: treshold is exceeded) Bit 3: Short to ground A (1: short condition deteted, driver currently shut down) Bit 4: Short to ground B (1: short condition detected, driver currently shut down) Bit 5: Open load A (1: no chopper event has happened during the last period with constant coil polarity) Bit 6: Open load B (1: no chopper event has happened during the last period with constant coil polarity) Bit 7: Stand still (1: no step impulse occurred on the step input during the last 2^20 clock cycles)	0/1	R
209	Encoder counter	The value of an encoder register can be read out or written.	[encoder steps]	RW
210	Encoder resolution	Resolution of the encoder in absolute positions. Quadrature encoder: 1 line = 4 positions	0… 65535 [positions]	RW
212	Max. encoder deviation	When the actual commanded position and the encoder position (parameter 209) differ more than defined by this parameter the motor will be stopped. This function is switched off when the maximum deviation is set to zero. If the value is negative and the maximum encoder deviation is exceeded, the motor current will be switched off so that the axis can be turned freely. The new starting position will be detected. With the next command the motor can be driven as usual. This function is used in open loop mode and closed loop mode.	−2.147.483.648… +2.147.483.647 [µsteps]	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
230	Gamma	Actual field weakening value (field weakening = speed dependent Back EMF compensation). The read-out value can be useful to choose values for axis parameters 108 and 109.	0… 255 Default = 255	R
233	Virtual actual position	With this parameter the actual virtual position of the ramp generator can be read out.	−2.147.483.648… +2.147.483.647 [µsteps]	R
236	CL actual target current scale factor	Actual target value of the current scale factor as defined by axis parameters 113, 114, 116, and 117. Due to the configurable delays using axis parameters 120 and 121, the actual target current scale factor may be different to the actual current scale factor.	0… 255 [1/256]	R

Number	Axis Parameter	Description	Range [Unit]	Acc.
237	Position error	This parameter indicates the difference between the virtual actual position of the ramp generator and the measured position of the motor.	−2.147.483.648… +2.147.483.647 [µsteps] [encoder steps]	R
254	Step/Dir mode	Please note, that there are differences using Step/Dir mode with *open loop* resp. *closed loop*. 0 – Turn OFF step/dir mode 1 – In *open loop mode*, use of the ENABLE input on the Step/Dir connector to switch between hold current and run current (no automatic switching). In *closed loop mode*, the ENABLE input can be used to switch the motor off resp. on. 2 – *Open loop mode*: automatic switching between hold and run current. After the first step pulse the module automatically switches over to run current, and a configurable time after the last step pulse the module automatically switches back to hold current. The ENABLE input is not used. *Closed loop mode*: maximum current. The ENABLE input is not used. 3 – Always use run current resp. maximum current. The ENABLE input is not used. 4 – O*pen loop mode*: automatic current switching like (2), but the ENABLE input is additionally used to switch the driver stage completely off or on. *Closed loop mode:* maximum current. The ENABLE input can be used to switch the driver stage completely on/off. 5 – *Closed loop mode* and *open loop mode*: always use run current resp. maximum current like (3), but the ENABLE input can be used to switch the driver stage completely off or on.	0… 5	RWE

6.1 Velocity Calculation

Some axis parameters are related to the speed of the motor. The unit of the velocity *<value>* is pulse per second (pps). For calculating the speed it is necessary to set the microstep resolution of the driver (axis parameter 140) first. Further, the fullsteps of the motor must be given. Now, calculate as follows:

$$rounds\ per\ second\ (rps) = \frac{< value >}{\text{microstep resolution of driver} * \text{fullsteps of motor}}$$

$$rounds\ per\ minute\ (rpm) = \text{rps} * 60$$

7 stallGuard2 Related Parameters

The module is equipped with a TMC262 motor driver chip. The TMC262 features load measurement that can be used for stall detection. stallGuard2 delivers a sensorless load measurement of the motor as well as a stall detection signal. The measured value changes linear with the load on the motor in a wide range of load, velocity and current settings. At maximum motor load the stallGuard2 value goes to zero. This corresponds to a load angle of 90° between the magnetic field of the stator and magnets in the rotor. This also is the most energy efficient point of operation for the motor.

Stall detection means that the motor will be stopped when the load gets too high. This level is set using axis parameter 174 (stallGuard2 threshold). In order to exclude e.g. resonances during motor acceleration and deceleration phases it is also possible to set a minimum speed for the motor being stopped due to stall detection using axis parameter 181.

Stall detection can also be used for finding the reference point. Do not use RFS in this case.

PARAMETERS NEEDED FOR ADJUSTING THE STALLGUARD2 FEATURE

<table>
<tr><th>Number</th><th>Axis Parameter</th><th>Description</th></tr>
<tr><td>6</td><td>absolute max. current
(CS / Current Scale)</td><td>The maximum value is 255. This value means 100% of the maximum current of the module. The current adjustment is within the range 0… 255 and can be adjusted in 32 steps.
<table>
<tr><td>0… 7</td><td>79…87</td><td>160… 167</td><td>240… 247</td><td rowspan="10">The most important motor setting, since too high values might cause motor damage!</td></tr>
<tr><td>8… 15</td><td>88… 95</td><td>168… 175</td><td>248… 255</td></tr>
<tr><td>16… 23</td><td>96… 103</td><td>176… 183</td><td></td></tr>
<tr><td>24… 31</td><td>104… 111</td><td>184… 191</td><td></td></tr>
<tr><td>32… 39</td><td>112… 119</td><td>192… 199</td><td></td></tr>
<tr><td>40… 47</td><td>120… 127</td><td>200… 207</td><td></td></tr>
<tr><td>48… 55</td><td>128… 135</td><td>208… 215</td><td></td></tr>
<tr><td>56… 63</td><td>136… 143</td><td>216… 223</td><td></td></tr>
<tr><td>64… 71</td><td>144… 151</td><td>224… 231</td><td></td></tr>
<tr><td>72… 79</td><td>152… 159</td><td>232… 239</td><td></td></tr>
</table></td></tr>
<tr><td>173</td><td>stallGuard2 filter enable</td><td>Enables the stallGuard2 filter for more precision of the measurement. If set, reduces the measurement frequency to one measurement per four fullsteps.
In most cases it is expedient to set the filtered mode before using coolStep.
Use the standard mode for step loss detection.
0 – standard mode
1 – filtered mode</td></tr>
<tr><td>174</td><td>stallGuard2 threshold</td><td>This signed value controls stallGuard2 threshold level for stall output and sets the optimum measurement range for readout. A lower value gives a higher sensitivity. Zero is the starting value. A higher value makes stallGuard2™ less sensitive and requires more torque to indicate a stall.
<table>
<tr><td>0</td><td>Indifferent value</td></tr>
<tr><td>1… 63</td><td>less sensitivity</td></tr>
<tr><td>-1… -64</td><td>higher sensitivity</td></tr>
</table></td></tr>
<tr><td>181</td><td>stop on stall</td><td>Below this speed motor will not be stopped. Above this speed motor will stop in case stallGuard2 load value reaches zero.</td></tr>
<tr><td>206</td><td>actual load value</td><td>Readout of the actual load value used for stall detection (stallGuard2).</td></tr>
</table>

In this chapter only basic axis parameters are mentioned which concern stallGuard2. The complete list of axis parameters in chapter 6 contains further parameters which offer more configuration possibilities.

8 Closed-Loop Operation Related Axis Parameter

The TMCM-1311 focuses on operating stepper motors in a closed loop using encoder feedback to prevent from motor stall, to adapt the current amplitude for saving energy and reducing heating, and for precise and fast positioning.

8.1 General Closed Loop Axis Parameters

All values for electrical period and step positions focus on a microstep resolution of 256 steps per full step. For a stepper motor with 200 steps (1.8°) this results in 51200 microsteps per revolution. When using an encoder with lower resolution the encoder position is automatically scaled to a resolution of 256 microsteps per full step.

Number	Axis parameter	Description	Units / Default	Acc.
128	Ramp mode	Automatically set when using ROR, ROL, MVP or SAP 14 commands. 0: Position mode. Steps are generated, when the parameters actual position and target position differ. Trapezoidal speed ramps are provided. When switching into this mode (using MVP ABS|REL|COORD), the motor will immediately start when there is a position difference. 1: Velocity mode. The motor will run continuously and the speed will be changed with constant (maximum) acceleration, if the parameter target speed is changed. 2: Torque mode. In closed loop mode only, the motor can be driven with constant torque. The velocity is only limited by the motor characteristics and the current setting. This mode of operation cannot be used in open loop mode!	0/1/2	RW
129	Closed loop operation	0: open loop mode 1: closed loop mode When switching from open loop to closed loop mode the closed loop system will be initialized. *Attention:* Wait for the closed loop init flag to be set to 1 after switching to closed loop mode (see axis parameter 133).	0/1	RW
133	CL init flag	0: open loop mode or closed loop not initialized 1: closed loop is initialized and ready for use See also parameter 129.	0/1	R
207	Max encoder deviation error flag	When maximum deviation is reached, motor is stopped/switched off. This flag shows this condition.	0/1	R

Number	Axis parameter	Description	Units / Default	Acc.
212	Max. encoder deviation	When the actual commanded position and the encoder position (parameter 209) differ more than defined by this parameter the motor will be stopped. This function is switched off when the maximum deviation is set to zero. If the value is negative and the maximum encoder deviation is exceeded, the motor current will be switched off so that the axis can be turned freely. The new starting position will be detected. With the next command the motor can be driven as usual. This function is used in open loop mode and closed loop mode.	−2.147.483.648… +2.147.483.647 [µsteps]	RWE

8.2 General Structure of the Closed Loop System

The general structure of the closed loop system of the TMCM-1311 focuses on ease of use. Basically, the stepper motor will be controlled similar to open loop mode. Nevertheless, extended functionality is provided for control of position, velocity, and current amplitude. The structure consists of three major blocks: a ramp generator, a torque control block, and the driver plus actuator. The structure is shown in Figure 8.1.

The **ramp generator** is primarily used for ramp calculation, velocity control (velocity mode) and position control (position mode). When using the TMCM-1311 in open loop, this block behaves just like a typical ramp generator and the stepper motor can lose steps if load is too high. In closed loop mode, the ramp generator provides extended functionality to control position and velocity.

The **torque control** block is used to control commutation angle and current level. Further it provides an option to compensate back EMF at higher speeds. Current level control can be switched off but closed loop operation is still possible. When using current level control the current amplitude is dynamically adapted to the actual load condition and the actual position error accounting for high drive efficiency and low heat dissipation in the stepper motor. Additionally, the torque control block is used without ramp generator for position maintenance during motor stand-still.

The **driver block** is realized using the integrated Trinamic TMC262 stepper motor pre-driver and a dedicated MOSFET stage. The TMC262 powers the stepper motor windings with the commanded current vectors of the torque control block. The actuator is the stepper motor itself including an encoder for position feedback. Rigid coupling and good alignment between motor and feedback is required for proper closed loop operation.

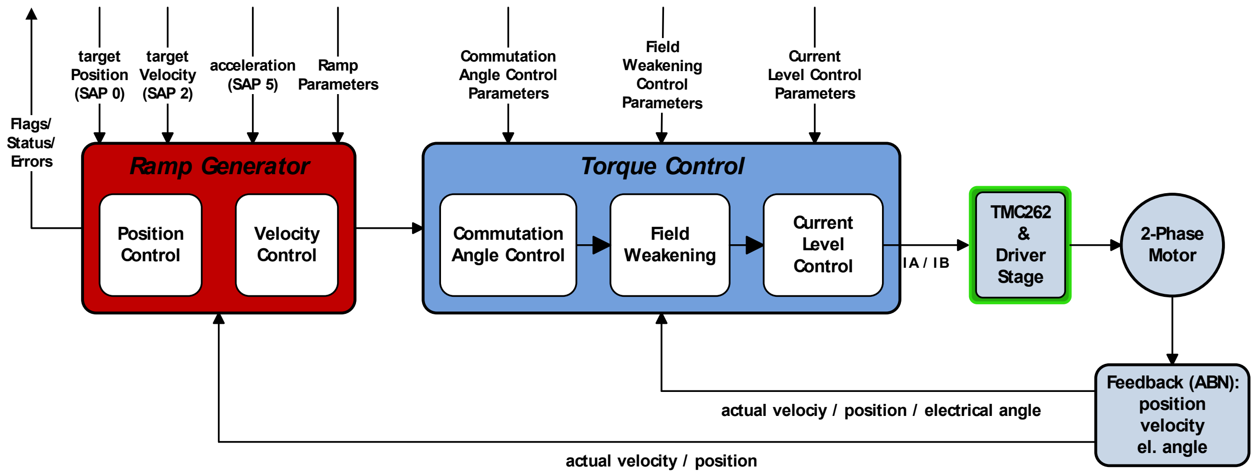


Figure 8.1 Closed Loop System Structure

8.3 Setting Encoder Resolution and Motor Resolution

Before starting closed loop operation, the resolution of the encoder and the motor must be configured properly. Therefore, set just two parameters: encoder resolution (axis parameter 210) and motor resolution (axis parameter 202).

ENCODER RESOLUTION

Encoders that have a binary resolution (e.g. 32768 steps per round) can be used as well as encoders with a decimal resolution (e.g. 40000 steps per round). To configure the resolution of the encoder use parameter 210 and input the encoder resolution directly. For a typical ABN type quadrature encoder, the resolution in number of positions or increments is the number of lines 4 times. E.g., if you have a 1000 lines encoder, the resolution is 1000 * 4 = 4000.

Note: SSI encoder interface is not yet implemented.

MOTOR RESOLUTION

For proper operation especially in closed loop mode, the motor resolution must also be configured. The default value is 200 as most stepper motors have 200 fullsteps with 1.8° each. If your motor is different, you need to adapt parameter 202 by simply configuring the number of fullsteps of the motor.

AXIS PARAMETERS RELATED TO ENCODER RESOLUTION AND MOTOR RESOLUTION

Number	Axis parameter	Description	Units / Default	Acc.
201	Encoder mode	Operation mode of the encoder. 4 (bit 2): Clear encoder on next null channel event. 8 (bit 3): Clear encoder on every null channel event. 16 (bit 4): Null channel polarity (active high when set)	4, 8, 16	RWE
202	Motor Resolution	Motor fullstep resolution.	0… 400 Default: 200 [fullsteps]	RW
210	Encoder resolution	Resolution of the encoder in absolute positions. Quadrature encoder: 1 line = 4 positions	0… 65535 [positions]	RW
112	CL Encoder Offset	Offset between encoder and electrical angle for correction of possible misalignment.	0 *default* [encoder steps]	RW

ATTENTION!

The encoder for a stepper motor with, e.g., 200 fullsteps per rotation should have a minimum encoder resolution of 9 bit in order to have adequate supporting points per fullstep. This is necessary to avoid step loss in case a fullstep change happens.

Minimum encoder resolution > fullsteps of stepper motor *2

8.4 Positioning Mode

In *positioning mode*, a certain target position (SAP 0) is set by the user and the TMCM-1311 uses the ramp profile parameters for acceleration (SAP 5) and target velocity (SAP 4) to make a ramp movement to the target position. This is similar to open loop operation. A start/stop velocity can be set (SAP 130) to start the ramp from a different velocity than zero.

In *closed loop mode*, the actual target position (during movement) and the final target position (as set by the user) are maintained and the motor does not lose steps or stalls. The dynamic behavior of this position maintenance can be adapted using various parameters.

Online Error Compensation

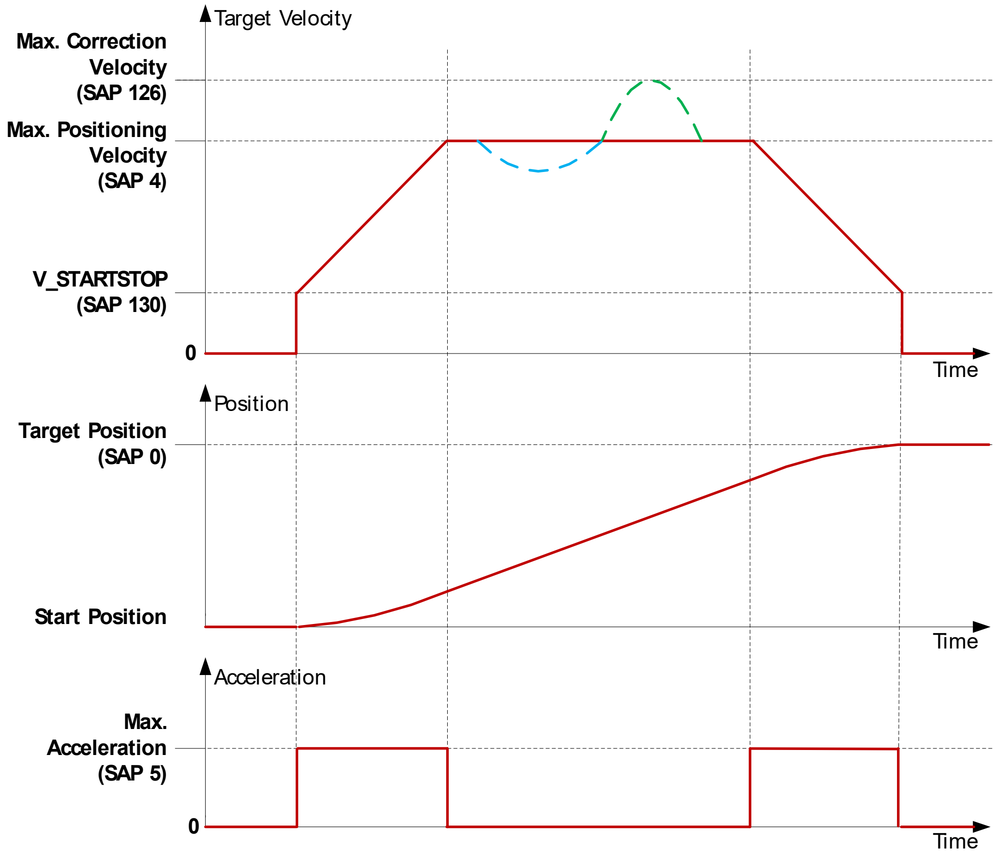


Figure 8.2 Typical Parameters for Positioning Mode

If the *target velocity* and the *actual target position* cannot be reached during movement due to high load (blue broken line in Figure 8.2), a *maximum correction velocity* (axis parameter 126) can be defined to compensate the position error between *virtual target position* of the ramp generator (axis parameter 233) and actual position of the drive (according to encoder feedback / *encoder counter* axis parameter 209).

The behavior of this online error compensation (green broken line in Figure 8.2) can be configured using a *correction velocity proportional factor* (axis parameter 124) and a configurable *maximum following error* (axis parameter 125) that is allowed before actually starting compensation.

If the *maximum correction velocity* is equal to or smaller than the *maximum positioning velocity*, the target position may not be reached in the time as defined by ramp parameters. Instead, the ramp motion will be extended until the target position is reached. In case it is desired to reach the target position just in time as specified by ramp parameters, TRINAMIC recommends setting a higher *maximum correction velocity*.

The ramp parameters can be changed on the fly during movement. The ramp will also be updated on-line.

EXAMPLE: ERROR COMPENSATION

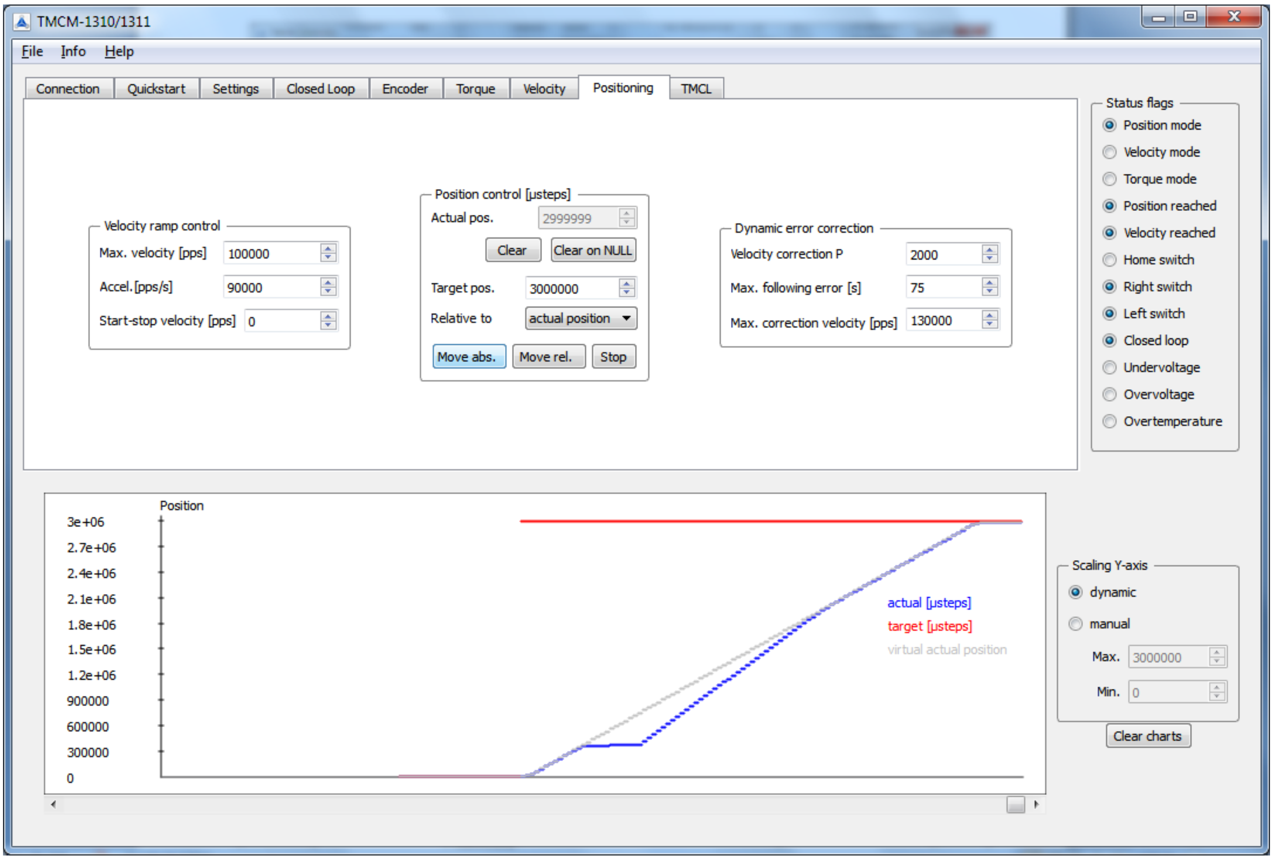


Figure 8.3 Example: online error compensation

In this example the target position has been reached in time though the load on the motor occasionally got too high. The closed loop error compensation had been able to make up leeway by executing dynamic error correction parameters. The virtual actual position (grey) calculated by the ramp generator and the actual position measured by the encoder (blue) merged at the end. Positioning has been carried out successfully.

AXIS PARAMETERS USED IN POSITIONING MODE

Number	Axis parameter	Description	Units / Default	Acc.
0	Target position	The desired position in position mode (see ramp mode, no. 138).	−2.147.483.648… +2.147.483.647 [encoder steps]	RW
1	Actual position	The current position of the motor.	−2.147.483.648… +2.147.483.647 [encoder steps]	RW
2	Target speed	The desired speed in velocity mode (see ramp mode / parameter 138). In position mode, this parameter is set by hardware: - during acceleration to maximum speed - during deceleration and rest to zero	-327.678.000… +327.679.999 [pps]	RW
3	Actual speed	The current rotation speed.	-327.678.000… +327.679.999 [pps]	RW
4	Maximum positioning speed	Should not exceed the physically highest possible value.	0… +327.679.999 [pps]	RWE
5	Maximum acceleration	The limit for acceleration and deceleration.	1… +24.999.998 [pps/s]	RW
8	Target pos. reached	Indicates that the actual position equals the target position.	0/1	R

Number	Axis parameter	Description	Units / Default	Acc.
124	CL correction velocity proportional factor	Proportional factor for on-line / live position lag compensation in positioning mode during a ramp movement. For a very quick compensation while the drive is active choose a high / the maximum value.	0… 5535	RW
125	CL max. following error	Maximum allowed following error during a ramp movement before starting compensation the position lag using parameters 124 and 126.	0… +268.435.454 Default = 0 [µsteps]	RW
126	CL max. correction velocity	Maximum correction speed during positioning mode. If a certain ramp/motion profile is used and a lag occurs during movement, the velocity will be increased to the maximum correction speed to compensate the position lag/ following error. If set to 0 or smaller than target velocity, the ramp profile will be simply extended if there is a lag between actual position and commanded position. If greater than target velocity the position lag will be compensated on-line / live.	0… +327.679.999 [pps]	RW
127	Relative positioning option	Positioning relative to one out of three starting points can be initialized using this parameter. 0 | last target position 1 | actual ramp generator position 2 | actual encoder position	0/1/2	RW
130	Start/Stop Velocity	Ramp generation for acceleration and deceleration begins/ends with this start and stop velocity value. When set to equal to the target speed, no ramp is generated. When set smaller than the target velocity (also to zero), the ramp starts with this velocity value. Must be smaller than target velocity (axis parameter 4).	0… +327.679.999 Default = 0 [pps]	RWE

8.5 Position Maintenance and Standstill Mode

When doing a ramp movement in positioning mode there are two parameters that can be used to configure the behavior for setting the *target_reached* flag:

- Axis parameter 134 defines a window around the target position in which the position is considered as reached.
- Additionally, parameter 135 defines the duration the motor must be within the position window defined by axis parameter 134 to set the *target_reached* flag.

Adapting these parameters prevents from long settling times around the target position. Tune for fast response times, e.g., for applications with fast positioning.

When the target position is reached and the *target_reached* flag has been set, the system is in ***standstill mode***.

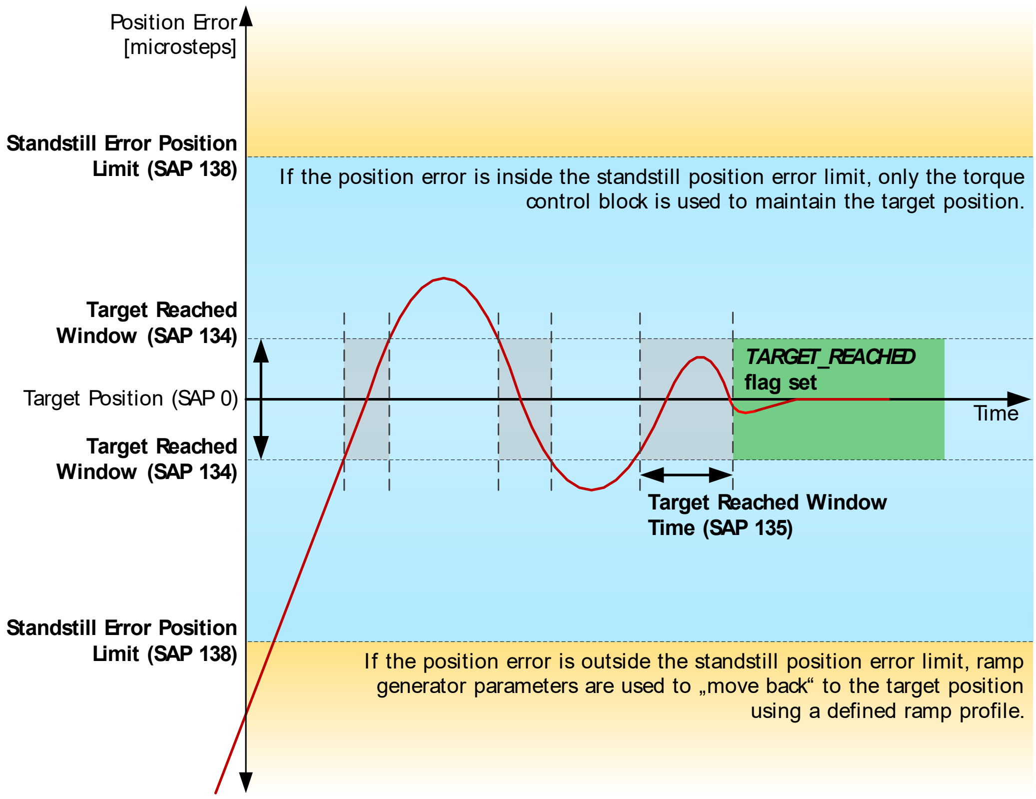


Figure 8.4 Principle and parameters for position maintenance

Position error inside the standstill position error limit (blue area)

If load is applied to the motor shaft in standstill mode the target position is maintained by the torque control block as long as the position error is within the range defined by axis parameter 138 (*standstill error position limit* / blue area in Figure 8.4). That is, the target position is maintained by adapting commutation angle and current amplitude. Thereby, the position error is amplified with a *gain factor* (axis parameter 136) for position maintenance. A higher gain factor provides higher stiffness. Additionally, a *dampening factor* (axis parameter 137) can be used to prevent from oscillations when using a high gain factor. More information on the current amplitude regulation is given in section 8.8.

Position error outside the standstill position error limit (orange area)

If the position error further increases and exceeds the limit defined by axis parameter 138, the ramp generator will be activated. In this case, the ramp generator uses the configured acceleration and velocity values to move back to the target position (orange areas).

AXIS PARAMETERS RELATED TO POSITIONING MAINTENANCE AND STANDSTILL MODE

Number	Axis parameter	Description	Units / Default	Acc.
8	Target pos. reached	Indicates that the actual position equals the target position.	0/1	R
134	CL position reached window	Window around the target position in which the target position will be considered as being reached. This value should be adjusted to the application and the resolution of the encoder. If the actual position is within the target reached window for at least or longer than defined by axis parameter 135, the *position_reached* flag will be set.	0… 255 [encoder steps] Default = 50	RW
135	CL position reached time	Minimum duration the motor must be within the target reached window (axis parameter 134) before the position will be considered as reached and *position_reached* flag will be set.	0… 131072 Default = 100 [1/10 ms]	RW
136	CL standstill position error gain	When the target position is reached, the velocity regulation will be switched off and the system is in a special standstill mode. In this mode, the position is hold and maintained as long as the position error is within the range of the standstill error limit (axis parameter 138) around the target position. This parameter is a gain factor for the position error used for position maintenance in standstill mode. - A value of 20 typically provides good results. - Higher values provide higher stiffness. - For values greater than 25 a dampening factor > 0 (axis parameter 137) should be used as well to prevent from oscillations.	10… 50 Default = 10	RW
137	CL standstill position error dampening factor	When the target position is reached, the velocity regulation is switched off and the system is in a standstill mode where the position is hold and maintained as long as the position error is within the range of the standstill error limit (axis parameter 138) around the target position. Parameter 137 is a dampening factor to prevent from oscillations when using a high proportional gain (axis parameter 136). - When axis parameter 136 is between 10 and 20, this parameter can be 0. - For higher proportional gain, e.g., 30, a dampening factor of 20 is a good starting value.	0… 65535 Default = 0	RW
138	CL standstill position error limit	When the target position is reached, the position maintenance by the ramp generator is switched off and the system is in a standstill mode. As long as the position error is within the range around the target position defined by axis parameter 138, the position maintenance is done using parameter 136 and 137. When the position error exceeds the range defined by this parameter, the configured ramp parameters are used to move back to the target position (as in normal positioning mode).	0… +268.435.454 [µsteps] Default=255	RW

8.6 Velocity Mode

In velocity mode, the motor is commanded to move at constant velocity. The target velocity is defined by axis parameter 2. Current level and phase angle will be adapted to maintain the commanded velocity. If desired, the ramp up can be made using a defined acceleration phase by setting axis parameter 5.
When the target velocity is reached (measured via encoder interface), the *velocity_reached* flag (GAP 16) is set. A *velocity reached window* can be defined using axis parameter 17. If the actual velocity is within this window the target velocity is considered as reached.

Number	Axis parameter	Description	Units / Default	Acc.
16	CL velocity reached	This flag is set when the actual velocity is within the velocity reached window (axis parameter 17) around the target position.	0/1	R
17	CL Velocity reached window	Window around the target velocity value in which the target velocity will be considered as being reached. The *velocity_reached* flag will be set accordingly.	0… +268.435.454 [pps]	RW
130	Start/Stop velocity	Ramp generation for acceleration and deceleration begins/ends with this start and stop velocity value. When set to equal to the target speed, no ramp is generated. When set smaller than the target velocity (also to zero), the ramp starts with this velocity value. Must be smaller than target velocity (axis parameter 4).	0… +327.679.999 Default = 0 [pps]	RWE

8.7 Torque Mode

In torque mode, the motor is commanded to run with fixed torque output. The velocity varies depending on the actual load. The motor runs as fast as it can for a certain load situation. In this mode, the commutation angle is always 90 degrees ahead for maximum torque output.

With axis parameter 14, the *target phase current amplitude* can be configured [mA] to the desired value. It is fixed as long as the motor is running in torque mode. The current amplitude will not be regulated. The range of axis parameter 14 covers the maximum current range allowed by the TMCM-1311, which is up to 4200mA peak phase current. The moving direction is defined by the sign of axis parameter 14.

HOW TO USE TRAPEZOID RAMPS IN TORQUE MODE:

- Set the *current slope* with axis parameter 20. This way, trapezoid ramps with acceleration and deceleration of the drive can be generated.
- Set a *start and stop value* for the ramp using axis parameter 141.
- To get the actual state of the ramp, the *actual current in torque mode* can be read out with axis parameter 19. This parameter can also be set to specify a value for the actual current, if necessary.

AXIS PARAMETERS RELATED TO TORQUE MODE

Number	Axis parameter	Description	Units / Default	Acc.
14	CL torque mode target current	Target RMS current value for torque mode. Positive and negative values define rotation direction. - Writing a target value to this parameter automatically switches to torque mode. - Reading provides the actual configured target current while in torque mode. - Reading while in other modes (velocity mode, position mode) provides information on the actual advance angle (in these cases the unit is microsteps). The maximum current that can be configured can be read out using axis parameter 15.	-3000… +3000 [mA]	RW *torque mode* R *velocity and position mode*
15	Maximum possible current	Based on axis parameters 6 and 179 this parameter returns the maximum possible RMS current.	0….+3000 [mA]	R
19	CL torque mode actual current	Actual current in torque mode	-3000… +3000 [mA]	RW
20	CL torque mode slope	Slope in torque mode (related to acceleration and deceleration).	[mA/s]	RW
141	CL torque mode start/stop current	Start and stop current in torque mode.	0… +3000 [mA]	RW
150	Motor current	Actual motor current $\sqrt{(AP151^2 + AP152^2)}$	[mA]	R
151	Current Phase A	Actual peak current Phase A	[mA]	R
152	Current Phase B	Actual peak current Phase B	[mA]	R
153	Supply Voltage	Actual value of supply voltage	[1/10 V]	R
154	DC Current	Actual DC current of the complete module + motor	[mA]	R
155	Module temperature	Actual temperature of the module	[°C]	R

8.8 Current Regulation

This section explains the *current* amplitude regulation of the closed loop system. Basically, the self-acting current regulation is used to improve drive efficiency by adapting the current level to a value that is just sufficient enough to move the load with the desired speed. To achieve this, the current amplitude is automatically adapted based on the actual position error and the configured parameters. This way, energy cost can be kept down and heat dissipation can be reduced.

In case this is not desired, the feature can be switched off completely using axis parameter 122 (*current scale enable*). Then, the motor is simply driven with the configured maximum current level. Nevertheless, it is still in closed loop mode and does not lose any steps or stall.

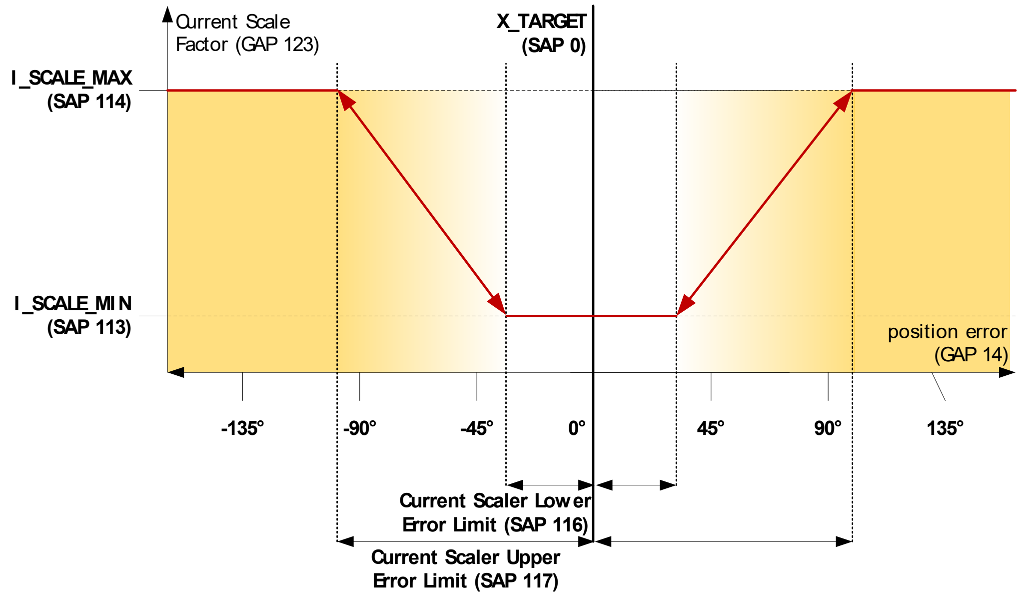


Figure 8.5 Basic principle of position error dependent current amplitude regulation

EXPLANATIONS RELATED TO FIGURE 8.5

Based on the position error with respect to the actual target position, the current amplitude (which is the length of the current vector of phase A and phase B current) is adapted. The red lines in Figure 8.5 show the position error dependent *target current scale factor*.

The target current amplitude is defined by four axis parameters as shown in Figure 8.5:

- Axis parameters 113 and 114 define the minimum and maximum current scale value, which are factors the maximum current (as defined by axis parameter 6) is scaled with.
- With axis parameters 116 and 117 two error limits can be configured. Within these two limits the current is scaled (increased or decreased) using a linear function.

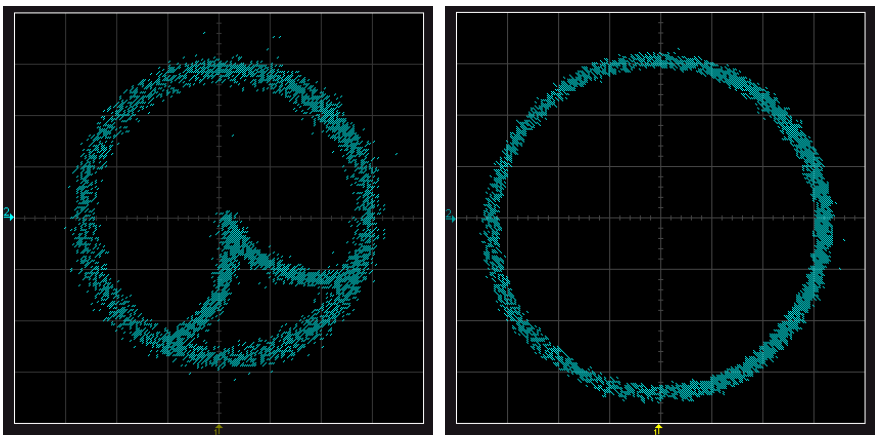


Figure 8.6 Basic principle of position error dependent current amplitude regulation

Explanations related to Figure 8.6

Figure 8.6 shows the two phase currents for phase A and B as XY-diagram on a scope using two current probes. They show the difference between *current regulator enabled* (left) and *disabled* (right). One complete circle represents a full electrical period of 360°. On the left side, current level regulation is switched on. When moving the rotor away from its target position (in this example somewhere in the lower right quadrant) the current amplitude is increased based on the position error up to its configured maximum amplitude. When approaching the target position the current level is decreased again. On the right side, the current regulation is switched off and current amplitude is always on the configured maximum regardless if there is a position error or not.

Additionally, there are a couple of parameters to adapt the dynamic behavior of the current regulation (shown in Figure 8.7):

- Axis parameters 118 and 119 configure the increment and decrement value for the current scale factor. Each time the current scale factor needs to be increased or decreased this value is added or subtracted from the actual value until the actual target current scale factor is reached.
- Axis parameters 120 and 121 define a delay timer value. Each time the current scale factor needs to be updated (increased or decreased) a timer is started with the respective delay value. The next increment or decrement will only be done when the timer is zero. If the parameters 120 and 121 are set to zero (default), the current scale factor is always directly set to the target current scale factor as defined by axis parameters 113, 114, 116, and 117. The effects of axis parameters 120 and 121 are shown in Figure 8.7. For example, the current scale value can immediately be set to its target value while it will be decreased with some delay.

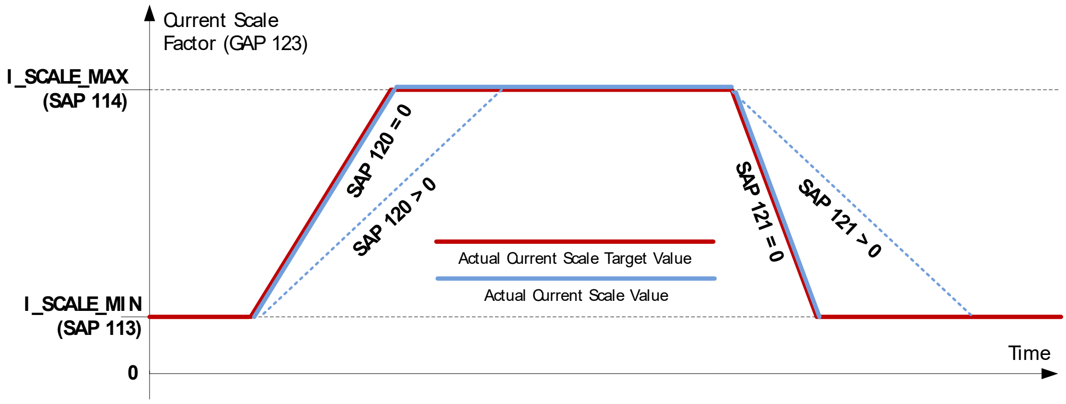


Figure 8.7 Using delay times for current increase and decrease

Axis parameters related to the current regulation

Number	Axis parameter	Description	Units / Default	Acc.
113	CL current scale minimum	Minimum current scale factor for current regulation. 255 = 1 = 100% of maximum current 127 = 0.5 = 50% of maximum current … *Attention!* The maximum current itself is defined by the CS parameter of the motor driver chip (see axis parameter 6)	Default = 15 [1/256]	RW

Number	Axis parameter	Description	Units / Default	Acc.
114	CL current scale maximum	Maximum current scale factor for current regulation. 255 = 1 = 100% of maximum current 127 = 0.5 = 50% of maximum current … *Attention!* - The maximum current itself is defined by the CS parameter of motor driver chip (see axis parameters 6) - The physical maximum current is defined by the sense resistors and MOSFETs of the TMCM-1311 and is limited to 3.0A RMS.	Default = 255 [1/256]	RW
115	CL current scale input select	Current scaling using raw position error or product of position error and gain factor (depends on axis parameters 136 and 137). 0 = only raw position error used for current scaling 1 = (position error * gain) used for current scaling	0/1 Default = 1	RW
116	CL current scale lower error limit	Position error from which on the current amplitude is increased (current scale factor is increased).	Default = 0 [encoder steps]	RW
117	CL current scale upper error limit	Position error from which on the current amplitude will be increased to its configured maximum (parameter 114). This parameter must be higher than axis parameter 116.	Default = 255 [encoder steps]	RW
118	CL current scale increment value	Current scale increment value if the actual current scale factor is below the calculated current scale factor target value. This parameter defines the step width at which the current scale factor will be increased.	Default = 1 [1/256]	RW
119	CL current scale decrement value	Current scale decrement value if the actual current scale factor is higher than the calculated current scale target value. This parameter defines the step width at which the current scale factor will be decreased.	Default = 1 [1/256]	RW
120	CL current scale increment timeout	This parameter defines the delay between two current scale factor increments and thereby controls the rate at which the current scale factor will be increased. Setting a timeout value here serves for dampening and prevents from high oscillations. 0 = the scale factor will directly be set to the actual target value without delay.	Default = 1 [ms]	RW
121	CL current scale decrement timeout	This parameter defines the delay between two current scale factor decrements and thereby controls the rate at which the current scale factor will be decreased, e.g., in order to prevent from oscillations around the target position. Setting a timeout value here serves for dampening and prevents from high oscillations. 0 = the scale factor will directly be set to the actual target value without delay.	Default = 1 [ms]	RW

Number	Axis parameter	Description	Units / Default	Acc.
122	CL current scale enable	1 = current scaling function on for closed loop 0 = current scaling function off, closed loop operation is still possible The current scaling functionality can be switched off if full specified current amplitude shall be used all the time. When switched on, the current scaling functionality adapts the current according to the configured profile. This saves energy and keeps the motor cooler.	Default = 1	RW
123	CL actual current scale factor	Actual value of the current scale factor.	0… 255 [1/256]	R
236	CL actual target current scale factor	Actual target value of the current scale factor as defined by axis parameters 113, 114, 116, and 117. Due to the configurable delays using axis parameters 120 and 121, the actual target current scale factor may be different to the actual current scale factor.	0… 255 [1/256]	R

8.9 Field Weakening

For higher velocities a correction of the advance angle is required to compensate for motor Back EMF and phase shift between electrical and magnetic field. Therefore, axis parameters 108 and 109 are used. These parameters define corner velocities at which the maximum commutation angle is allowed to go beyond 90 degrees up to a maximum of 180 degrees (= 2 motor fullsteps). The increase is linearly between the field weakening minimum and the maximum velocity.

Adjusting the field weakening parameters 108 and 109 allows for automatic motor back EMF compensation in case high velocities are needed. TRINAMIC recommends to start with default values and adjust the field weakening minimum and maximum velocities carefully. Actual gamma/field weakening values can be read out on the fly using axis parameter 230.

This mechanism is also applied in positioning and torque mode when exceeding the configured corner velocities.

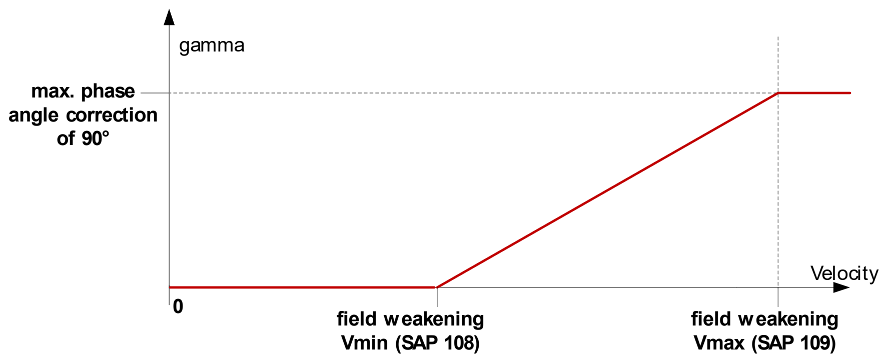


Figure 8.8 Principle and parameters for field weakening in velocity mode

AXIS PARAMETERS RELATED TO FIELD WEAKENING

Number	Axis parameter	Description	Units / Default	Acc.
108	CL field weakening minimum velocity	Minimum motor speed at which the speed dependent Back EMF compensation will be applied (field weakening). Based on the velocity measured via encoder feedback.	0… +327.679.999 [pps]	RW
109	CL field weakening maximum velocity	Maximum motor speed for the speed dependent Back EMF compensation will be applied (field weakening). Based on the velocity measured via encoder feedback.	0… +327.679.999 [pps]	RW
230	Gamma	Actual field weakening value (field weakening = speed dependent Back EMF compensation). *This read-out value can be useful to choose values for axis parameters 108 and 109.*	0… 255	R

8.10 Status and Feedback Information

Some parameters can be read out, e.g., for visualization and feedback.

AXIS PARAMETERS FOR READOUT

Number	Axis parameter	Description	Units / Default
8	Target pos. reached	Indicates that the actual position equals the target position.	0/1
15	Maximum possible current	Based on axis parameters 6 and 179 this parameter returns the maximum possible RMS current.	0… +3000 [mA]
16	CL velocity reached	This flag is set when the actual velocity is within the velocity reached window (axis parameter 17) around the target position.	0/1
19	CL torque mode actual current	Actual current in torque mode.	-3000… +3000 [mA]
123	CL actual current scale factor	Actual value of the current scale factor.	0… 255 [1/256]
236	CL actual target current scale factor	Actual target value of the current scale factor as defined by axis parameters 113, 114, 116, and 117. Due to the configurable delays using axis parameters 120 and 121, the actual target current scale factor may be different to the actual current scale factor.	0… 255 [1/256]
133	CL init flag	0: open loop mode or closed loop not initialized 1: closed loop is initialized and ready for use See also axis parameter 129.	0/1
150	Motor current	Actual motor current $\sqrt{(AP151^2 + AP152^2)}$	[mA]
151	Current Phase A	Actual peak current Phase A	[mA]
152	Current Phase B	Actual peak current Phase B	[mA]
153	Supply Voltage	Actual value of supply voltage	[1/10 V]
154	DC Current	Actual DC current of the complete module + motor	[mA]
155	Module temperature	Actual temperature of the module	[°C]
230	Gamma	Actual field weakening value (field weakening = speed dependent commutation angle compensation). The read-out value can be useful to choose values for axis parameters 108 and 109.	0… 2.147.483.647 [pps]
233	Virtual actual position	With this parameter the actual virtual position of the ramp generator can be read out.	−2.147.483.648… +2.147.483.647 [µsteps]
236	CL actual target current scale factor	Actual target value of the current scale factor as defined by axis parameters 113, 114, 116, and 117. Due to the configurable delays using axis parameters 120 and 121, the actual target current scale factor may be different to the actual current scale factor.	0… 255 [1/256]
237	Position error	This parameter indicates the difference between the virtual actual position of the ramp generator and the measured position of the motor.	−2.147.483.648… +2.147.483.647 [encoder steps]

8.11 Example Programs: Closed Loop Operation

Both example programs show that it is necessary to set the maximum current first. Then, check (example 2) or set (example 1) the motor steps and the encoder steps before switching to closed loop operation. *After setting axis parameter 126 to 1 for closed loop mode wait until the closed loop flag is set!*

8.11.1 Example Program 1

To run example program 1, an encoder with 40000 increments per rotation has to be connected. For positioning, the commands MVP ABS and WAIT POS are used.

```
//Initialize the parameters
          SAP 6, 0, 128         //max. current
          SAP 202, 0, 200       //motor full steps
          SAP 210, 0, 40000     //encoder steps

          SAP 4, 0, 398105      //positioning speed
          SAP 5, 0, 3981070     //acceleration
          SAP 126, 0, 1023950   //correction speed

          SAP 129, 0, 1         //switch on closed loop
ClWait:   GAP 133, 0            //wait until closed loop is ready
          JC ZE, ClWait

//Let motor run
Loop:     MVP ABS, 0, 512000
          WAIT POS, 0, 0
          WAIT TICKS, 0, 100
          MVP ABS, 0, 0
          WAIT POS, 0, 0
          WAIT TICKS, 0, 100
          JA Loop
          STOP
```

8.11.2 Example Program 2

In the second example program, the encoder resolution is detected automatically. Here, the motor will run increasing and decreasing the positioning counter while using position reached interrupts.

```
        SAP 6, 0, 128    //set max. motor current

        //Start encoder detection and wait until ready
        SAP 132, 0, 1
Detect: GAP 132, 0
        COMP 2
        JC NE, Detect

        //Switch on closed loop and wait until ready
        SAP 129, 0, 1
Init:   GAP 133, 0
        JC ZE, Init

        //Let motor run (using position reached interrupt)
        VECT 3, PosReached1
        EI 3
        EI 255

        MVP ABS, 0, 51200
Loop:   WAIT TICKS, 0, 1000
        JA Loop


PosReached1:
        VECT 3, PosReached2
        MVP ABS, 0, 0
        RETI

PosReached2:
        VECT 3, PosReached1
        MVP ABS, 0, 51200
        RETI
```

9 Reference Search

The built-in reference search features switching point calibration using the limit switches or an optional home switch. The home switch can be connected to input IN1. Switches with normally closed (NC) contacts are used for the limit switches and for the home switches. The internal operation is based on a state machine that can be started, stopped and monitored (instruction RFS, no. 13).

HINTS FOR REFERENCE SEARCH:

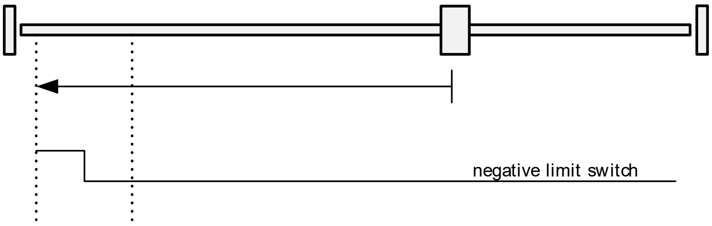
- The settings of the automatic stop functions corresponding to the switches (axis parameters 12 and 13) have no influence on the reference search.
- Until the reference switch is found for the first time, the speed given by axis parameter 194 is used for searching the switch.
- After hitting the switch, the motor slowly moves until the switch is released. Finally the switch is re-entered in the other direction, setting the reference point to the center of the two switching points. This low calibrating speed is given by axis parameter 195.
- Axis parameter 193 selects the reference search mode that is to be used.

PARAMETERS NEEDED FOR REFERENCE SEARCH

Number	Axis Parameter	Description
9	Ref. switch status	The logical state of the reference (left) switch. See the TMC 429 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.
10	Right limit switch status	The logical state of the (right) limit switch.
11	Left limit switch status	The logical state of the left limit switch (in three switch mode)
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.
193	Ref. search mode	1 search left stop switch only 2 search right stop switch, then search left stop switch 3 search right stop switch, then search left stop switch from both sides 4 search left stop switch from both sides 5 search home switch in negative direction, reverse the direction when left stop switch reached 6 search home switch in positive direction, reverse the direction when right stop switch reached 7 search home switch in positive direction, ignore end switches 8 search home switch in negative direction, ignore end switches *Adding 128 to these values reverses the polarity of the home switch input.*
194	Referencing search speed	For the reference search this value directly specifies the search speed.
195	Referencing switch speed	Similar to parameter no. 194, the speed for the switching point calibration can be selected.
196	Distance end switches	This parameter provides the distance between the end switches after executing the RFS command (mode 2 or 3).

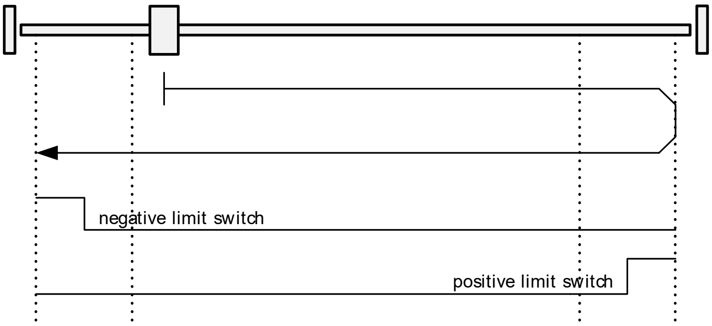
9.1.1 Reference Search Modes (Axis Parameter 193)

SAP 193, 0, 1



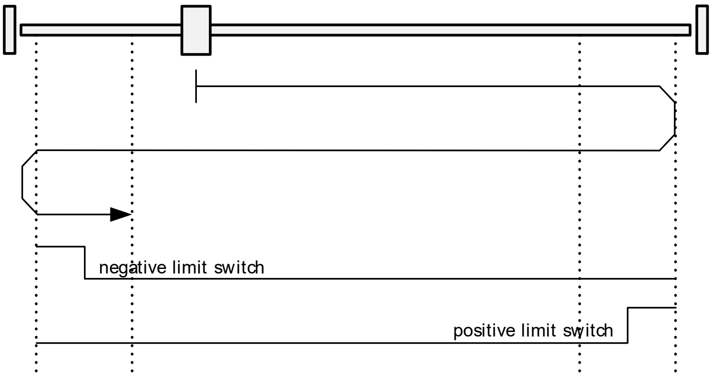
Search left stop switch only.

SAP 193, 0, 2



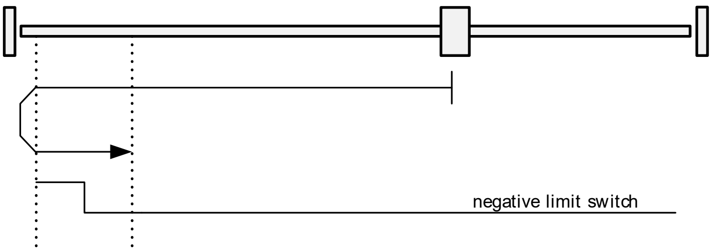
Search right stop switch, then search left stop switch.

SAP 193, 0, 3



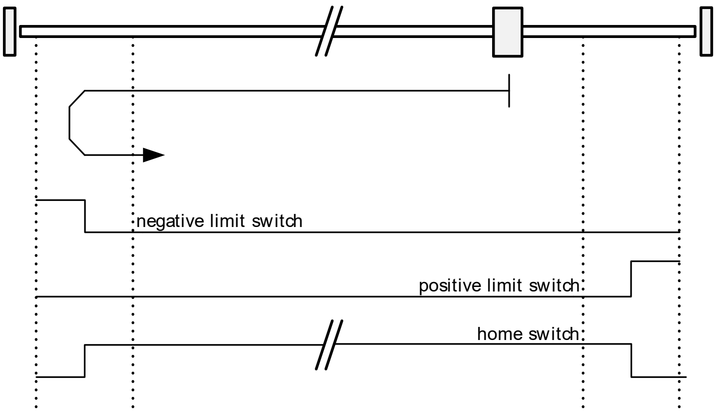
Search right stop switch, then search left stop switch from both sides.

SAP 193, 0, 4



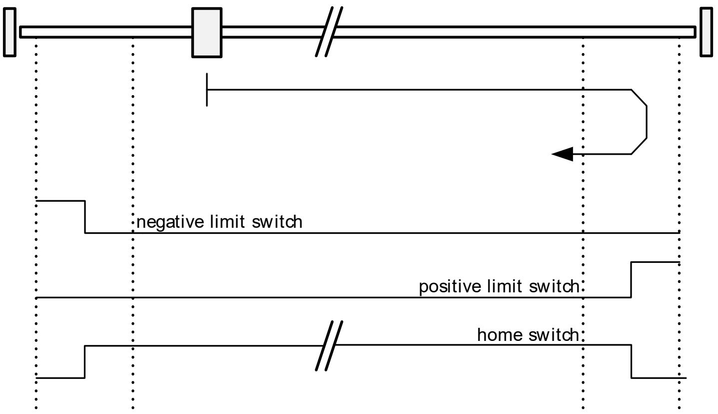
Search left stop switch from both sides.

SAP 193, 0, 5



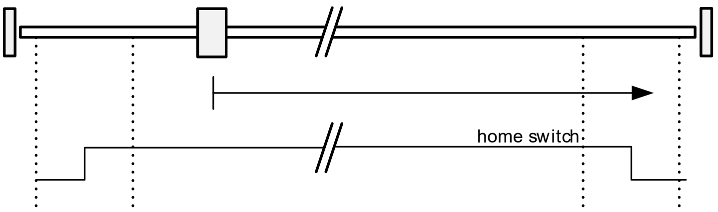
Search home switch in negative direction, reverse the direction when left stop switch reached.

SAP 193, 0, 6



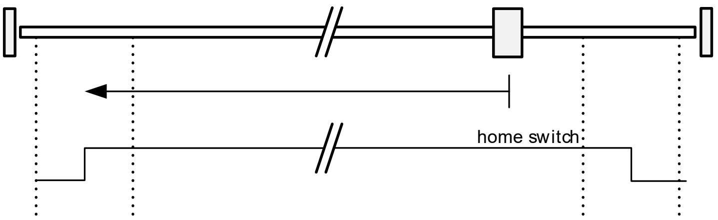
Search home switch in positive direction, reverse the direction when right stop switch reached.

SAP 193, 0, 7



Search home switch in positive direction, ignore end switches.

SAP 193, 0, 8



Search home switch in negative direction, ignore end switches.

Figure 9.1 Reference search modes

10 Global Parameters

GLOBAL PARAMETERS ARE GROUPED INTO 4 BANKS:

- bank 0 (global configuration of the module)
- bank 1 (user C variables)
- bank 2 (user TMCL variables)
- bank 3 (interrupt configuration)

Please use SGP and GGP commands to write and read global parameters.

10.1 Bank 0

PARAMETERS 64… 132

Parameters with numbers from 64 on configure stuff like *auto start mode* or *end switch polarity*. Change these parameters to meet your needs. The best and easiest way to do this is to use the appropriate functions of the TMCL-IDE. The parameters with numbers between 64 and 128 are stored in EEPROM only.

An SGP command on such a parameter will always store it permanently and no extra STGP command is needed. Take care when changing these parameters, and use the appropriate functions of the TMCL-IDE to do it in an interactive way.

MEANING OF THE LETTERS IN COLUMN *ACCESS*:

Access type	Related command(s)	Description
R	GGP	Parameter readable
W	SGP, AGP	Parameter writable
E	STGP, RSGP	Parameter stored permanently in EEPROM

Number	Global parameter	Description	Range	Access
64	EEPROM magic	Setting this parameter to a different value as $E4 will cause re-initialization of the axis and global parameters (to factory defaults) after the next power up. This is useful in case of miss-configuration.	0… 255	RWE
65	RS485 baud rate	0 – 9600 baud – *Default* 1 – 14400 baud 2 – 19200 baud 3 – 28800 baud 4 – 38400 baud 5 – 57600 baud 6 – 76800 baud – *Not supported by Windows!* 7 – 115200 baud 8 – 230400 baud 9 – 250000 baud – *Not supported by Windows!* 10 – 500000 baud – *Not supported by Windows!* 11 – 1000000 baud – *Not supported by Windows!*	0… 11	RWE
66	Serial address	The module (target) address for RS485.	0… 255	RWE
67	ASCII mode	Configure the TMCL ASCII interface: Bit 0: 0 – start up in binary (normal) mode 1 – start up in ASCII mode Bits 4 and 5: 00 – Echo back each character 01 – Echo back complete command 10 – Do not send echo, only send command reply		RWE
68	Serial heartbeat	Serial heartbeat for the RS485 interface. If this time limit is up and no further command is noticed the motor will be stopped. 0 – parameter is disabled	[ms]	RWE

Number	Global parameter	Description	Range	Access
69	CAN bit rate	2 20kBit/s 3 50kBit/s 4 100kBit/s 5 125kBit/s 6 250kBit/s 7 500kBit/s 8 1000kBit/s *Default*	2… 8	RWE
70	CAN reply ID	The CAN ID for replies from the board (default: 2)	0… 7ff	RWE
71	CAN ID	The module (target) address for CAN (default: 1)	0… 7ff	RWE
73	configuration EEPROM lock flag	Write: 1234 to lock the EEPROM, 4321 to unlock it. Read: 1=EEPROM locked, 0=EEPROM unlocked.	0/1	RWE
75	Telegram pause time	Pause time before the reply via RS485 is sent. For RS485 it is often necessary to set it to 15 (for RS485 adapters controlled by the RTS pin). For CAN interface this parameter has no effect!	0… 255	RWE
76	Serial host address	Host address used in the reply telegrams sent back via RS485.	0… 255	RWE
77	Auto start mode	0: Do not start TMCL application after power up (default). 1: Start TMCL application automatically after power up.	0/1	RWE
79	End switch polarity	0: normal polarity 1: reverse polarity	0/1	RWE
81	TMCL code protection	Protect a TMCL program against disassembling or overwriting. 0 – no protection 1 – protection against disassembling 2 – protection against overwriting 3 – protection against disassembling and overwriting ***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
83	CAN secondary address	Second CAN ID for the module. Switched off when set to zero.	0… 7ff	RWE
84	Coordinate storage	0 – coordinates are stored in the RAM only (but can be copied explicitly between RAM and EEPROM) 1 – coordinates are always stored in the EEPROM only	0 or 1	RWE
85	do not restore user variables	0 – user variables are restored (*default*) 1 – user variables are not restored	0/1	RWE
88	Interface selection	0 – CAN interface 1 – RS485 interface	0/1	RWE
128	TMCL application status	0 –stop 1 – run 2 – step 3 – reset	0… 3	R
129	Download mode	0 – normal mode 1 – download mode	0/1	R
130	TMCL program counter	The index of the currently executed TMCL instruction.		R
132	Tick timer	A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value.		RW
133	Random number	Choose a random number. ***Read only!***	0… 2147483647	R

10.2 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 5.7.36) these variables form the interface between extensions of the firmware (written in C) and TMCL applications.

10.3 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*:

Access type	Related command(s)	Description
R	GGP	Parameter readable
W	SGP, AGP	Parameter writable
E	STGP, RSGP	Parameter stored permanently in EEPROM

Number	Global parameter	Description	Range	Access
0… 55	general purpose variable #0… #55	for use in TMCL applications	-2^{31}… $+2^{31}$	RWE
56… 255	general purpose variables #56… #255	for use in TMCL applications	-2^{31}… $+2^{31}$	RW

10.4 Bank 3

Bank 3 contains interrupt parameters. Some interrupts need configuration (e.g. the timer interval of a timer interrupt). This can be done using the SGP commands with parameter bank 3 (SGP <type>, 3, <value>).

The priority of an interrupt depends on its number. Interrupts with a lower number have a higher priority.

The following table shows all interrupt parameters that can be set.

MEANING OF THE LETTERS IN COLUMN *ACCESS*:

Access type	Related command(s)	Description
R	GGP	Parameter readable
W	SGP, AGP	Parameter writable
E	STGP, RSGP	Parameter stored permanently in EEPROM

Number	Global parameter	Description	Range	Access
0	Timer 0 period (ms)	Time between two interrupts (ms)	32 bit unsigned [ms]	RW
1	Timer 1 period (ms)	Time between two interrupts (ms)	32 bit unsigned [ms]	RW
2	Timer 2 period (ms)	Time between two interrupts (ms)	32 bit unsigned [ms]	RW
27	Stop left 0 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW
28	Stop right 0 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW
39	Input 0 edge type	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW
40	Input 1 edge type	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW
41	Input 2 edge type	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW
42	Input 3 edge type	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW
43	Input 4 edge type	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW
44	Input 5 edge type	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW
45	Input 6 edge type	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW
46	Input 7 edge type	0=off, 1=low-high, 2=high-low, 3=both	0… 3	RW

11 TMCL Programming Techniques and Structure

11.1 Initialization

The first task in a TMCL program (like in other programs also) is to initialize all parameters where different values than the default values are necessary. For this purpose, SAP and SGP commands are used.

11.2 Main Loop

Embedded systems normally use a main loop that runs infinitely. This is also the case in a TMCL application that is running stand alone. Normally the auto start mode of the module should be turned on. After power up, the module then starts the TMCL program, which first does all necessary initializations and then enters the main loop, which does all necessary tasks end never ends (only when the module is powered off or reset).

There are exceptions to this, e.g. when TMCL routines are called from a host in direct mode.

So most (but not all) standalone TMCL programs look like this:

```
//Initialization
      SAP 4, 0, 500  //define max. positioning speed
      SAP 5, 0, 100  //define max. acceleration

MainLoop:
      //do something, in this example just running between two positions
      MVP ABS, 0, 5000
      WAIT POS, 0, 0
      MVP ABS, 0, 0
      WAIT POS, 0, 0
      JA MainLoop       //end of the main loop => run infinitely
```

11.3 Using Symbolic Constants

To make your program better readable and understandable, symbolic constants should be taken for all important numerical values that are used in the program. The TMCL-IDE provides an include file with symbolic names for all important axis parameters and global parameters.

EXAMPLE

```
//Define some constants
#include TMCLParam.tmc
MaxSpeed = 500
MaxAcc = 100
Position0 = 0
Position1 = 5000

//Initialization
      SAP APMaxPositioningSpeed, Motor0, MaxSpeed
      SAP APMaxAcceleration, Motor0, MaxAcc

MainLoop:
      MVP ABS, Motor0, Position1
      WAIT POS, Motor0, 0
      MVP ABS, Motor0, Position0
      WAIT POS, Motor0, 0
      JA MainLoop
```

Just have a look at the file TMCLParam.tmc provided with the TMCL-IDE. It contains symbolic constants that define all important parameter numbers.

Using constants for other values makes it easier to change them when they are used more than once in a program. You can change the definition of the constant and do not have to change all occurrences of it in your program.

11.4 Using Variables

The *User Variables* can be used if variables are needed in your program. They can store temporary values. The commands SGP, GGP and AGP are used to work with user variables:

SGP is used to set a variable to a constant value (e.g. during initialization phase).

GGP is used to read the contents of a user variable and to copy it to the accumulator register for further usage.

AGP can be used to copy the contents of the accumulator register to a user variable, e.g. to store the result of a calculation.

EXAMPLE

```
MyVariable = 42
      //Use a symbolic name for the user variable
      //(This makes the program better readable and understandable.)

SGP MyVariable, 2, 1234   //Initialize the variable with the value 1234
...
...
GGP MyVariable, 2         //Copy the contents of the variable to the
accumulator register
CALC MUL, 2                //Multiply accumulator register with two
AAP MyVariable, 2         //Store contents of the accumulator register to the
variable
...
...
```

Furthermore, these variables can provide a powerful way of communication between a TMCL program running on a module and a host. The host can change a variable by issuing a direct mode SGP command (remember that while a TMCL program is running direct mode commands can still be executed, without interfering with the running program). If the TMCL program polls this variable regularly it can react on such changes of its contents.

The host can also poll a variable using GGP in direct mode and see if it has been changed by the TMCL program.

11.5 Using Subroutines

The *CSUB* and *RSUB* commands provide a mechanism for using subroutines. The *CSUB* command branches to the given label. When an *RSUB* command is executed the control goes back to the command that follows the *CSUB* command that called the subroutine.

This mechanism can also be nested. From a subroutine called by a *CSUB* command other subroutines can be called. In the current version of TMCL eight levels of nested subroutine calls are allowed.

11.6 Mixing Direct Mode and Standalone Mode

Direct mode and stand alone mode can also be mixed. When a TMCL program is being executed in standalone mode, direct mode commands are also processed (and they do not disturb the flow of the program running in standalone mode). So, it is also possible to query e.g. the actual position of the motor in direct mode while a TMCL program is running.

Communication between a program running in standalone mode and a host can be done using the TMCL user variables. The host can then change the value of a user variable (using a direct mode SGP command) which is regularly polled by the TMCL program (e.g. in its main loop) and so the TMCL program can react on such changes. Vice versa, a TMCL program can change a user variable that is polled by the host (using a direct mode GGP command).

A TMCL program can be started by the host using the run command in direct mode. This way, also a set of TMCL routines can be defined that are called by a host. In this case it is recommended to place JA commands at the beginning of the TMCL program that jump to the specific routines. This assures that the entry addresses of the routines will not change even when the TMCL routines are changed (so when changing the TMCL routines the host program does not have to be changed).

EXAMPLE

```
//Jump commands to the TMCL routines
Func1:      JA Func1Start
Func2:      JA Func2Start
Func3:      JA Func3Start

Func1Start: MVP ABS, 0, 1000
            WAIT POS, 0, 0
            MVP ABS, 0, 0
            WAIT POS, 0, 0
            STOP

Func2Start: ROL 0, 500
            WAIT TICKS, 0, 100
            MST 0
            STOP

Func3Start:
            ROR 0, 1000
            WAIT TICKS, 0, 700
            MST 0
            STOP
```

This example provides three very simple TMCL routines. They can be called from a host by issuing a run command with address 0 to call the first function, or a run command with address 1 to call the second function, or a run command with address 2 to call the third function. You can see the addresses of the TMCL labels (that are needed for the run commands) by using the *Generate symbol file* function of the TMCL-IDE.

Please refer to the TMCL-IDE User Manual for further information about the TMCL-IDE.

12 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.

13 Revision History

13.1 Firmware Revision

Version	Date	Author	Description
1.02	2012-SEP-27	OK	First version
1.06beta	2013-MAY-23	OK	Preliminary version
1.07	2014-JUNE-02	OK	Release version
1.08		OK	Not deployed
1.09		OK	Not deployed
1.10	2013-OKT-10	OK	- Correction related to positioning window. - Correction related to condition for switching into position hold mode. - Correction related to current regulation in case current scale limits are outside 90˚. - The following axis parameters are new or updated: 14, 18, 19, 20, 141, and 212. - Attention: torque mode has new features related to ramp generation.
1.11	2014-MAR-17	OK	Enhancements - Unit for line item specification resp. positioning in closed loop mode: encoder steps instead of microsteps. - I^2t control added (axis parameters 25-29). - I^2t status flag added. - The value of axis parameter 112 (encoder offset) can be stored now. Corrections - Actual position value in open loop mode can be changed now. - The USB interface reports the correct USB version now.
1.12	2014-DEC-19	OK	- Internal position offset overflow problems eliminated. - In closed-loop mode speed and acceleration now also given in encoder steps instead of motor microsteps. - GIO 255, 0 command fixed.
1.13	2016-MAY-25	OK	- Using new Trinamic USB IDs. - RS485 Telegram Pause Time (SGP 75) fixed. - Interface selection (SGP 88) fixed. - End switch functionality fixed.
1.14	2016-NOV-23	OK	- GIO 4, 0 command fixed.

13.2 Document Revision

Version	Date	Author	Description
1.00	2012-DEC-05	SD	First version
1.10	2013-MAY-23	SK	Closed Loop Section and Parameters updated including tables and figures. Preliminary Version
1.11	2013-JUN-14	SK, OK	Additions/changes for release version
1.12	2013-JUL-03	SD	Axis parameters updated. Changes related to wording and design. Chapter 8 completed.
1.13	2013-JUL-05	SD	New front picture
1.14	2013-OKT-22	SD	- Descriptions in chapter 8.6 velocity mode updated. - Descriptions in chapter 8.7 updated. - The following axis parameters are new or updated: 14, 18, 19, 20, 141, and 212. - Chapter 8.3 (encoder resolution) updated. - Attention: torque mode has new features related to ramp generation.

1.15	2013-10-31	SD	Axis parameters 4 and 5 corrected.
1.16	2014-MAR-19	SD	- Unit for line item specification resp. positioning in closed loop mode: encoder steps instead of microsteps. - Axis parameter 254 deleted. - I^2t control (axis parameters 25-29) added. - I^2t status flag description added. - Axis parameter 112 description updated. - Value ranges of velocity related axis parameters updated.
1.17	2015-NOV-19	JP	- Description of global parameter 88 enhanced.
1.18	2018-MAR-01	OK	- Chapter 9 revised. - Covers firmware V1.14.

14 References

[TMCM-1311] TMCM-1311 Hardware Manual
[TMCL-IDE] TMCL-IDE User Manual

Please refer to www.trinamic.com.