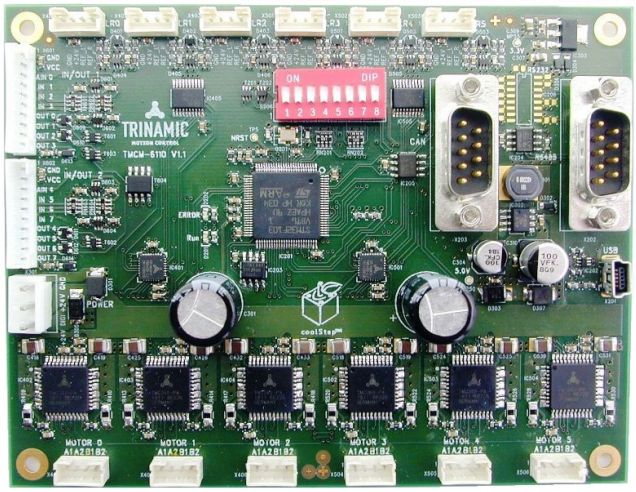
TMCM-6110 TMCL™ Firmware Manual

Firmware Version V1.47 | Document Revision V1.23 • 2021-NOV-02

The TMCM-6110 is a six axis controller/driver module for 2-phase bipolar stepper motors. The TMCM-6110 TMCL firmware allows to control the module using TMCL™ commands, supporting standalone operation as well as direct mode control, making use of the Trinamic TMC429 motion controller and the TMC262 motor driver. Dynamic current control, and quiet, smooth and efficient operation are combined with StallGuard2™ and CoolStep™ features.



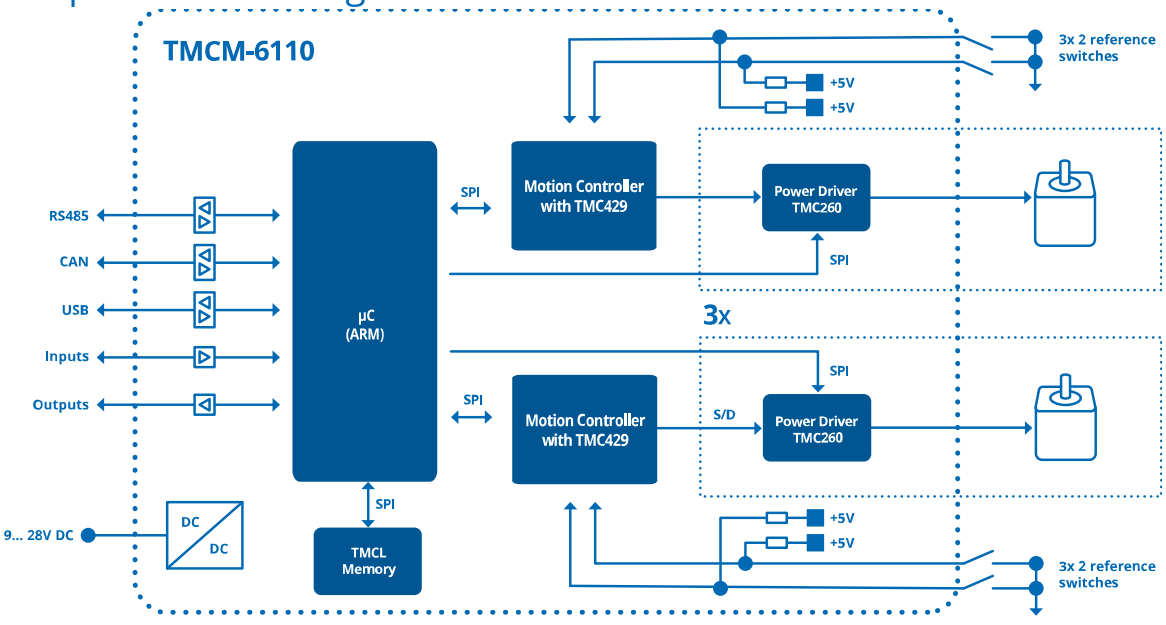
Features

- Six axis stepper motor control
- Supply voltage 9…28V DC
- TMCL™
- USB interface
- RS485 interface
- CAN interface
- CoolStep™
- StallGuard2™

Applications

- Laboratory Automation
- Manufacturing
- Semiconductor Handling
- Robotics
- Factory Automation
- Test & Measurement
- Life Science
- Biotechnology
- Liquid Handling

Simplified Block Diagram



©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at: www.trinamic.com

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

ADI Trinamic™

Read entire documentation.

Contents

1 Features **5**
1.1 StallGuard2 . . . 6
1.2 CoolStep . . . 6

2 First Steps with TMCL **7**
2.1 Basic Setup . . . 7
2.2 Using the TMCL Direct Mode . . . 7
2.3 Changing Axis Parameters . . . 7
2.4 Testing with a simple TMCL Program . . . 8

3 TMCL and the TMCL-IDE — An Introduction **10**
3.1 Binary Command Format . . . 10
3.1.1 Checksum Calculation . . . 11
3.2 Reply Format . . . 12
3.2.1 Status Codes . . . 12
3.3 Standalone Applications . . . 13
3.4 The ASCII Interface . . . 14
3.4.1 Entering and leaving the ASCII Mode . . . 14
3.4.2 Format of the Command Line . . . 14
3.4.3 Format of a Reply . . . 15
3.4.4 Configuring the ASCII Interface . . . 15
3.5 TMCL Command Overview . . . 16
3.6 TMCL Commands by Subject . . . 18
3.6.1 Motion Commands . . . 18
3.6.2 Parameter Commands . . . 18
3.6.3 Branch Commands . . . 19
3.6.4 I/O Port Commands . . . 19
3.6.5 Calculation Commands . . . 20
3.6.6 Interrupt Processing Commands . . . 20
3.6.7 New TMCL Commands . . . 23
3.7 Detailed TMCL Command Descriptions . . . 25
3.7.1 ROR (Rotate Right) . . . 25
3.7.2 ROL (Rotate Left) . . . 26
3.7.3 MST (Motor Stop) . . . 27
3.7.4 MVP (Move to Position) . . . 28
3.7.5 SAP (Set Axis Parameter) . . . 31
3.7.6 GAP (Get Axis Parameter) . . . 32
3.7.7 STAP (Store Axis Parameter) . . . 33
3.7.8 RSAP (Restore Axis Parameter) . . . 34
3.7.9 SGP (Set Global Parameter) . . . 35
3.7.10 GGP (Get Global Parameter) . . . 36
3.7.11 STGP (Store Global Parameter) . . . 37
3.7.12 RSGP (Restore Global Parameter) . . . 38
3.7.13 RFS (Reference Search) . . . 39
3.7.14 SIO (Set Output) . . . 41
3.7.15 GIO (Get Input) . . . 43
3.7.16 CALC (Calculate) . . . 46
3.7.17 COMP (Compare) . . . 48
3.7.18 JC (Jump conditional) . . . 49
3.7.19 JA (Jump always) . . . 51
3.7.20 CSUB (Call Subroutine) . . . 52
3.7.21 RSUB (Return from Subroutine) . . . 53

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.22 WAIT (Wait for an Event to occur) . . . 54
3.7.23 STOP (Stop TMCL Program Execution – End of TMCL Program) . . . 56
3.7.24 SCO (Set Coordinate) . . . 57
3.7.25 GCO (Get Coordinate) . . . 58
3.7.26 CCO (Capture Coordinate) . . . 60
3.7.27 ACO (Accu to Coordinate) . . . 61
3.7.28 CALCX (Calculate using the X Register) . . . 62
3.7.29 AAP (Accu to Axis Parameter) . . . 64
3.7.30 AGP (Accu to Global Parameter) . . . 65
3.7.31 CLE (Clear Error Flags) . . . 66
3.7.32 EI (Enable Interrupt) . . . 68
3.7.33 DI (Disable Interrupt) . . . 69
3.7.34 VECT (Define Interrupt Vector) . . . 70
3.7.35 RETI (Return from Interrupt) . . . 72
3.7.36 CALCVV (Calculate using two User Variables) . . . 73
3.7.37 CALCVA (Calculate using a User Variable and the Accumulator Register) . . . 75
3.7.38 CALCAV (Calculate using the Accumulator Register and a User Variable) . . . 77
3.7.39 CALCVX (Calculate using a User Variable and the X Register) . . . 79
3.7.40 CALCXV (Calculate using the X Register and a User Variable) . . . 81
3.7.41 CALCV (Calculate using a User Variable and a Direct Value) . . . 83
3.7.42 RST (Restart) . . . 85
3.7.43 DJNZ (Decrement and Jump if not Zero) . . . 86
3.7.44 CALL (Conditional Subroutine Call) . . . 87
3.7.45 MVPA (Move to Position specified by Accumulator Register) . . . 89
3.7.46 ROLA (Rotate Left using the Accumulator Register) . . . 91
3.7.47 RORA (Rotate Right using the Accumulator Register) . . . 92
3.7.48 MVPXA (Move to Position specified by Accumulator Register and X Register) . . . 93
3.7.49 ROLXA (Rotate Left using the Accumulator Register and X Register) . . . 95
3.7.50 RORXA (Rotate Right using the Accumulator Register and X Register) . . . 96
3.7.51 MSTX (Motor Stop using X Register) . . . 97
3.7.52 SAPX (Set Axis Parameter, Axis specified by X Register) . . . 98
3.7.53 GAPX (Get Axis Parameter, Axis specified by X Register) . . . 99
3.7.54 AAPX (Accu to Axis Parameter, Axis specified by X Register) . . . 100
3.7.55 SIV (Set Indexed Variable) . . . 102
3.7.56 GIV (Get Indexed Variable) . . . 103
3.7.57 AIV (Accumulator to Indexed Variable) . . . 104
3.7.58 Customer specific Command Extensions (UF0...UF7 – User Functions) . . . 105
3.7.59 Request Target Position reached Event . . . 106
3.7.60 TMCL Control Commands . . . 108

4 Axis Parameters 110

5 Global Parameters 118
5.1 Bank 0 . . . 118
5.2 Bank 1 . . . 121
5.3 Bank 2 . . . 121
5.4 Bank 3 . . . 122

6 Module Specific Hints 124
6.1 Velocity and Acceleration Calculation . . . 124
6.1.1 Velocity Conversion . . . 124
6.1.2 Acceleration Conversion . . . 124
6.1.3 How to choose Pulse Divisor and Ramp Divisor . . . 125
6.1.4 Conversion between PPS, RPM and RPS . . . 125

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

7 Hints and Tips **126**
7.1 Reference Search . . . 126
7.1.1 Mode 1 . . . 127
7.1.2 Mode 2 . . . 127
7.1.3 Mode 3 . . . 127
7.1.4 Mode 4 . . . 128
7.1.5 Mode 5 . . . 128
7.1.6 Mode 6 . . . 129
7.1.7 Mode 7 . . . 129
7.1.8 Mode 8 . . . 130
7.2 StallGuard2 . . . 131
7.3 CoolStep . . . 132
7.4 Secondary Serial Address . . . 135
7.5 Secondary CAN Address . . . 135

8 TMCL Programming Techniques and Structure **136**
8.1 Initialization . . . 136
8.2 Main Loop . . . 136
8.3 Using Symbolic Constants . . . 136
8.4 Using Variables . . . 137
8.5 Using Subroutines . . . 138
8.6 Combining Direct Mode and Standalone Mode . . . 138
8.7 Make the TMCL Program start automatically . . . 139

9 Figures Index **140**

10 Tables Index **141**

11 Supplemental Directives **142**
11.1 Producer Information . . . 142
11.2 Copyright . . . 142
11.3 Trademark Designations and Symbols . . . 142
11.4 Target User . . . 142
11.5 Disclaimer: Life Support Systems . . . 142
11.6 Disclaimer: Intended Use . . . 142
11.7 Collateral Documents & Tools . . . 143

12 Revision History **144**
12.1 Firmware Revision . . . 144
12.2 Document Revision . . . 145

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

1 Features

The TMCM-6110 is a six axis controller/driver module for 2-phase bipolar stepper motors with state of the art feature set. It is highly integrated, offers a convenient handling and can be used in many decentralized applications. The module has been designed for coil currents up to 1.1A RMS and 28V DC supply voltage. It is also equipped with end switch inputs as well as some general purpose digital inputs and outputs and two analog inputs. Furthermore it is equipped with three ABN encoder inputs. With its high energy efficiency from TRINAMIC's CoolStep™ technology cost for power consumption is kept down. The TMCL firmware allows for both standalone and direct mode operation.

Main characteristics

- Motion controller & stepper motor driver:
  - Hardware motion profile calculation in real-time.
  - On the fly alteration of motion parameters (e.g. position, velocity, acceleration).
  - High performance microcontroller for overall system control and communication protocol handling.
  - Up to 256 microsteps per full step.
  - High-efficient operation, low power dissipation.
  - Dynamic current control.
  - Integrated protection.
  - StallGuard2™ feature for stall detection.
  - CoolStep™ feature for reduced power consumption and heat dissipation.
- Interfaces
  - USB interface.
  - RS485 bus.
  - CAN bus.
  - Eight additional digital inputs and eight additional digital outputs.
  - Two analog inputs.
  - End switch inputs for each axis.

Software

TMCL remote controlled operation via USB, RS485 or CAN interface and/or stand-alone operation via TMCL programming. PC-based application development software TMCL-IDE available for free.

Electrical data

- Supply voltage: 9V…28V supply range.
- Motor current: up to 1.1A RMS / 1.6A peak (programmable).

Please see also the separate Hardware Manual.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

1.1 StallGuard2

StallGuard2 is a high-precision sensorless load measurement using the back EMF of the coils. It can be used for stall detection as well as other uses at loads below those which stall the motor. The StallGuard2 measurement value changes linearly over a wide range of load, velocity, and current settings. At maximum motor load, the value reaches zero or is near zero. This is the most energy-efficient point of operation for the motor.

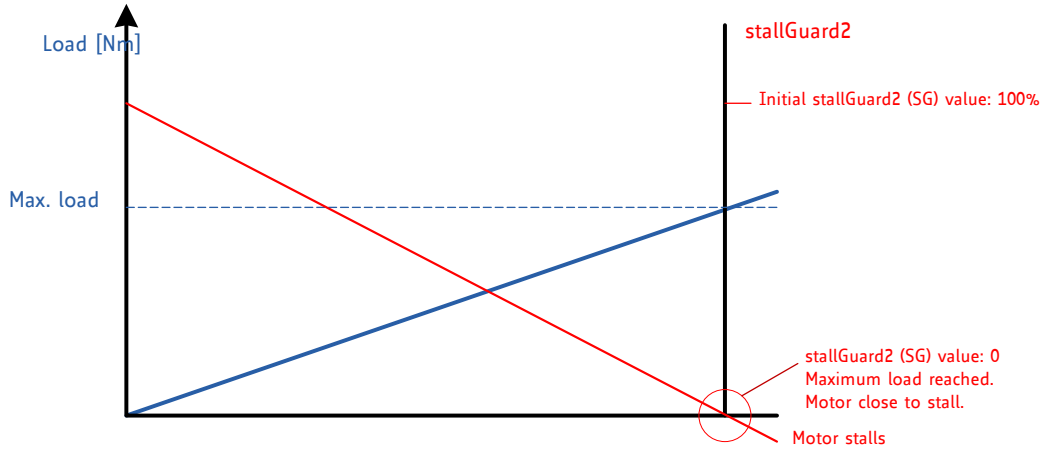


Figure 1: StallGuard2 Load Measurement as a Function of Load

1.2 CoolStep

CoolStep is a load-adaptive automatic current scaling based on the load measurement via StallGuard2 adapting the required current to the load. Energy consumption can be reduced by as much as 75%. CoolStep allows substantial energy savings, especially for motors which see varying loads or operate at a high duty cycle. Because a stepper motor application needs to work with a torque reserve of 30% to 50%, even a constant-load application allows significant energy savings because CoolStep automatically enables torque reserve when required. Reducing power consumption keeps the ystem cooler, increases motor life, and allows cost reduction.

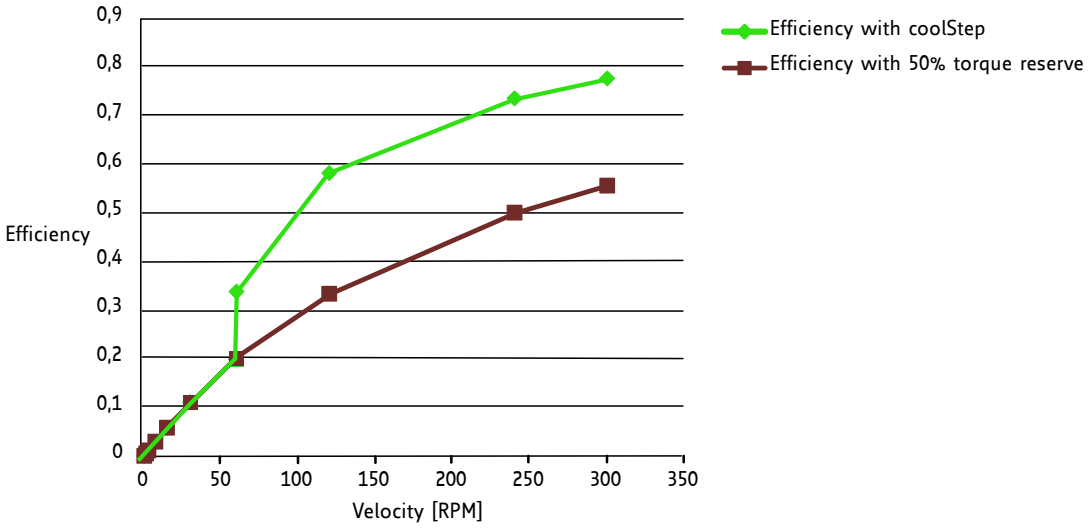


Figure 2: Energy Efficiency Example with CoolStep

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

2 First Steps with TMCL

In this chapter you can find some hints for your first steps with the TMCM-6110 and TMCL. You may skip this chapter if you are already familiar with TMCL and the TMCL-IDE.

Things that you will need

- Your TMCM-6110 module.
- A USB cable with micro USB plug or an RS485 interface with suitable cable or a CAN interface with suitable cable.
- A power supply (24V DC) for your TMCM-6110 module.
- The TMCL-IDE 3.x already installed on your PC
- A two-phase bipolar stepper motor.

2.1 Basic Setup

First of all, you will need a PC with Windows (at least Windows 7) and the TMCL-IDE 3.x installed on it. If you do not have the TMCL-IDE installed on your PC then please download it from the TMCL-IDE product page of Trinamic's website (http://www.trinamic.com) and install it on your PC.

Please also ensure that your TMCM-6110 is properly connected to your power supply and that the stepper motor is properly connected to the module. Please see the TMCM-6110 hardware manual for instructions on how to do this. **Do not connect or disconnect a stepper motor to or from the module while the module is powered!**
Then, please start up the TMCL-IDE. After that you can connect your TMCM-6110. The easiest way to do that is via USB. Just connect the module via USB and switch on the power supply for the module (while the TMCL-IDE is running on your PC). The module will be recognized by the TMCL-IDE, and necessary driver registrations in Windows will automatically done by the TMCL-IDE.

Of course you can also use RS485 or CAN bus. Please see the TMCM-6110 hardware manual on how to connect to RS485 or CAN bus.

2.2 Using the TMCL Direct Mode

At first try to use some TMCL commands in direct mode. In the TMCL-IDE a tree view showing the TMCM-6110 and all tools available for it is displayed. Click on the Direct Mode entry of the tool tree. Now, the Direct Mode tool will pop up.

In the Direct Mode tool you can choose a TMCL command, enter the necessary parameters and execute the command. For example, choose the command ROL (rotate left). Then choose the appropriate motor (motor 0 if your motor is connected to the motor 0 connector). Now, enter the desired speed. Try entering 51200 (pps) as the value and then click the Execute button. The motor will now run. Choose the MST (motor stop) command and click Execute again to stop the motor.

2.3 Changing Axis Parameters

Next you can try changing some settings (also called axis parameters) using the SAP command in direct mode. Choose the SAP command. Then choose the parameter type and the motor number. Last, enter the desired value and click execute to execute the command which then changes the desired parameter.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

The following table points out the most important axis parameters. Please see chapter 4 for a complete list of all axis parameters.

Most important axis parameters				
Number	Axis Parameter	Description	Range [Units]	Access
4	Maximum positioning speed	The maximum speed used for positioning ramps.	1...2047 [int]	RW
5	Maximum acceleration	Maximum acceleration in positioning ramps. Acceleration and deceleration value in velocity mode.	1...2047 [int]	RW
6	Maximum current	Motor current used when motor is running. The maximum value is 255 which means 100% of the maximum current of the module. *The most important setting, as too high values can cause motor damage.*	0...255	RW
7	Standby current	The current used when the motor is not running. The maximum value is 255 which means 100% of the maximum current of the module. This value should be as low as possible so that the motor can cool down when it is not moving. Please see also parameter 214.	0...255	RW

Table 1: Most important Axis Parameters

2.4 Testing with a simple TMCL Program

Now, test the TMCL stand alone mode with a simple TMCL program. To type in, assemble and download the program, you will need the TMCL creator. This is also a tool that can be found in the tool tree of the TMCL-IDE. Click the TMCL creator entry to open the TMCL creator. In the TMCL creator, type in the following little TMCL program:

```
    ROL 0, 1000                 //Rotate motor 0 with speed 1000
    WAIT TICKS, 0, 500
    MST 0
    ROR 0, 100                  //Rotate motor 0 with 1000
    WAIT TICKS, 0, 500
    MST 0

    SAP 4, 0, 100                //Set max. Velocity
    SAP 5, 0, 100                //Set max. Acceleration
Loop:
    MVP ABS, 0, 512000           //Move to Position 512000
    WAIT POS, 0, 0               //Wait until position reached
    MVP ABS, 0, -512000          //Move to Position -512000
    WAIT POS, 0, 0               //Wait until position reached
    JA Loop                      //Infinite Loop
```

After you have done that, take the following steps:

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

1. Click the Assemble icon (or choose Assemble from the TMCL menu) in the TMCL creator to assemble the program.
2. Click the Download icon (or choose Download from the TMCL menu) in the TMCL creator to donwload the program to the module.
3. Click the Run icon (or choose Run from the TMCL menu) in the TMCL creator to run the program on the module.

Also try out the debugging functions in the TMCL creator:

1. Click on the Bug icon to start the debugger.
2. Click the Animate button to see the single steps of the program.
3. You can at any time pause the program, set or reset breakpoints and resume program execution.
4. To end the debug mode click the Bug icon again.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3 TMCL and the TMCL-IDE — An Introduction

As with most TRINAMIC modules the software running on the microprocessor of the TMCM-6110 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-6110 supports TMCL direct mode (binary commands). It also implements standalone TMCL program execution. This makes it possible to write TMCL programs using the TMCL-IDE and store them in the memory of the module.

In direct mode the TMCL communication over RS-232, RS-485, CAN, and 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-6110. 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 the interface 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.

3.1 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 RS-232, RS-485, RS-422 or USB interface, it has to be enclosed by an address byte at the beginning and a checksum byte at the end. In these cases it consists of nine bytes.

The binary command format with RS-232, RS-485, RS-422 and USB is as follows:

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

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

Table 2: TMCL Command Format

Info	The checksum is calculated by accumulating all the other bytes using an 8-bit addition.

Note	When using the CAN interface, leave out the address byte and the checksum byte. With CAN, the CAN-ID is used as the module address and the checksum is not needed because CAN bus uses hardware CRC checking.

3.1.1 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 which show how to do this:

Checksum calculation 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
```

Checksum calculation 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
```

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.2 Reply Format

Every time a command has been sent to a module, the module sends a reply. The reply format with RS-232, RS-485, RS-422 and USB is as follows:

TMCL Reply Format	
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

Table 3: TMCL Reply Format

Info	The checksum is also calculated by adding up all the other bytes using an 8-bit addition. Do not send the next command before having received the reply!

Note	When using CAN interface, the reply does not contain an address byte and a checksum byte. With CAN, the CAN-ID is used as the reply address and the checksum is not needed because the CAN bus uses hardware CRC checking.

3.2.1 Status Codes

The reply contains a status code. The status code can have one of the following values:

TMCL Status Codes	
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

Table 4: TMCL Status Codes

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.3 Standalone Applications

The module is equipped with a TMCL memory for storing TMCL applications. You can use the 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.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.4 The ASCII Interface

There is also an ASCII interface that can be used to communicate with the module and to send some direct mode commands as text strings. Only 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, UF7.

Note	The ASCII interface is deprecated. It should not be used for new applications.

Note	Only direct mode commands can be entered in ASCII mode.

Note	The TMCL-IDE does not support communicating with the module in ASCII mode. So, in order to be able to use the TMCL-IDE with a module the module must be in binary mode. We normally recommend using the binary mode as this has some advantages over the ASCII mode. The ASCII mode is only provided here for compatibility. For new applications it is strongly recommended to use the binary mode.

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

- BIN: This command quits ASCII mode and returns to binary mode.
- RUN: This command can be used to start a TMCL program stored in memory.
- STOP: Stops a TMCL program which is currently running on the module.

3.4.1 Entering and leaving the ASCII Mode

- The ASCII command line interface is entered by sending the binary command 139 (enter ASCII mode).
- Afterwards the commands can be entered in their mnemonic form (e.g. via a terminal program).
- For leaving the ASCII mode and re-entering the binary mode enter the command BIN.

3.4.2 Format of the Command Line

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

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

The last command line shown above will make the module return to binary mode.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.4.3 Format of a Reply

After executing the command the module sends back a reply in ASCII format which consists of the following things:

- 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 which means that the command has been executed successfully.

3.4.4 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 5).

- Bit 0 determines the startup mode: if this bit is set, the module will start up in ASCII mode, else the module 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 will be echoed back when the module is addressed. Characters 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.

Note	When trying the ASCII mode for the first time it is strongly recommended not to use global parameter 67 but the direct mode command 139 to enter the ASCII mode. Then the module can also be switched back to binary mode by a power cycle if there should be trouble with the ASCII mode communication.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.5 TMCL Command Overview

This section gives a short overview of all TMCL commands.

Overview of all TMCL Commands			
Command	Number	Parameter	Description
ROR	1	<motor number>, <velocity>	Rotate right with specified velocity
ROL	2	<motor number>, <velocity>	Rotate left with specified velocity
MST	3	<motor number>	Stop motor movement
MVP	4	ABS|REL|COORD, <motor number>, <position|offset>	Move to position (absolute or relative)
SAP	5	<parameter>, <motor number>, <value>	Set axis parameter (motion control specific settings)
GAP	6	<parameter>, <motor number>	Get axis parameter (read out motion control specific settings)
STAP	7	<parameter>, <motor number>, <value>	Store axis parameter (store motion control specific settings)
RSAP	8	<parameter>, <motor number>	Restore axis parameter (restore motion control specific settings)
SGP	9	<parameter>, <bank number>, <value>	Set global parameter (module specific settings e.g. communication settings or TMCL user variables)
GGP	10	<parameter>, <bank number>	Get global parameter (read out module specific settings e.g. communication settings or TMCL user variables)
STGP	11	<parameter>, <bank number>	Store global parameter (TMCL user variables only)
RSGP	12	<parameter>, <bank number>	Restore global parameter (TMCL user variables only)
RFS	13	<START|STOP|STATUS>, <motor number>	Reference search
SIO	14	<port number>, <bank number>, <value>	Set digital output to specified value
GIO	15	<port number>, <bank number>	Get value of analog/digital input
CALC	19	<operation>, <value>	Aithmetical operation between accumulator and direct value
COMP	20	<value>	Compare accumulator with value
JC	21	<condition>, <jump address>	Jump conditional
JA	22	<jump address>	Jump absolute
CSUB	23	<subroutine address>	Call subroutine
RSUB	24		Return from subroutine

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Command	Number	Parameter	Description
EI	25	<interrupt number>	Enable interrupt
DI	26	<interrupt number>	Disable interrupt
WAIT	27	<condition>, <motor number>, <ticks>	Wait with further program execution
STOP	28		Stop program execution
SCO	30	<coordinate number>, <motor number>, <position>	Set coordinate
GCO	31	<coordinate number>, <motor number>	Get coordinate
CCO	32	<coordinate number>, <motor number>	Capture coordinate
CALCX	33	<operation>	Arithmetical operation between accumulator and X-register
AAP	34	<parameter>, <motor number>	Accumulator to axis parameter
AGP	35	<parameter>, <bank number>	Accumulator to global parameter
CLE	36	<flag>	Clear an error flag
VECT	37	<interrupt number>, <address>	Define interrupt vector
RETI	38		Return from interrupt
ACO	39	<coordinate number>, <motor number>	Accu to coordinate
CALCVV	40	<operation>, <user variable 1>, <user variable 2>	Arithmetical operation between two user variables
CALCVA	41	<operation>, <user variable>	Arithmetical operation between user variable and accumulator
CALCAV	42	<operation>, <user variable>	Arithmetical operation between accumulator and user variable
CALCVX	43	<operation>, <user variable>	Arithmetical operation between user variable and X register
CALCXV	44	<operation>, <user variable>	Arithmetical operation between X register and user variable
CALCV	45	<operation>, <value>	Arithmetical operation between user variable and direct value
MVPA	46	ABS|REL|COORD, <motor number>	Move to position specified by accumulator
MVPXA	47	ABS|REL|COORD, <motor number>	Move to position specified by accumulator, motor specified by X register
RST	48	<jump address>	Restart the program from the given address

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Command	Number	Parameter	Description
DJNZ	49	<user variable>, <jump address>	Decrement and jump if not zero
ROLA	50	<motor number>	Rotate left, velocity specified by accumulator
RORA	51	<motor number>	Rotate right, velocity specified by accumulator
ROLXA	52		Rotate left, velocity speciifed by accumulator, motor specified by X register
RORXA	53		Rotate right, velocity speciifed by accumulator, motor specified by X register
MSTX	54		Stop motor specified by X register
SIV	55	<value>	Set indexed variable
GIV	56		Get indexed variable
AIV	57		Accumulator to indexed variable

Table 5: Overview of all TMCL Commands

3.6 TMCL Commands by Subject

3.6.1 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.

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

Table 6: Motion Commands

3.6.2 Parameter Commands

These commands are used to set, read and store axis parameters or global parameters. Axis parameters can be set independently for each axis, whereas global parameters control the behavior of the module

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

itself. These commands can also be used in direct mode and in standalone mode.

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

Table 7: Parameter Commands

3.6.3 Branch Commands

These commands are used to control the program flow (loops, conditions, jumps etc.). Using them in direct mode does not make sense. They are intended for standalone mode only.

Branch Commands		
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

Table 8: Branch Commands

3.6.4 I/O Port Commands

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

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

I/O Port Commands		
Mnemonic	Command number	Meaning
SIO	14	Set output
GIO	15	Get input

Table 9: I/O Port Commands

3.6.5 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.

Calculation Commands		
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

Table 10: Calculation Commands

For calculating purposes there is an accumulator (also called 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.

Please see also chapter 3.6.7 for more calculation commands.

3.6.6 Interrupt Processing Commands

TMCL also contains functions for a simple way of interrupt processing. Using interrupts, many tasks can be programmed in an easier way.
The following commands are use to define and handle interrupts:

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

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

Table 11: Interrupt Processing Commands

3.6.6.1 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 include file Interrupts.inc in order to have symbolic constants for the interrupt numbers. Table 12 show all interrupts that are available on the TMCM-6110.

Interrupt Vectors	
Interrupt number	Interrupt type
0	Timer 0
1	Timer 1
2	Timer 2
3	Target position reached 0
4	Target position reached 1
5	Target position reached 2
6	Target position reached 3
7	Target position reached 4
8	Target position reached 5
15	StallGuard axis 0
16	StallGuard axis 1
17	StallGuard axis 2
18	StallGuard axis 3
19	StallGuard axis 4
20	StallGuard axis 5
27	Left stop switch 0
28	Right stop switch 0
29	Left stop switch 1
30	Right stop switch 1
31	Left stop switch 2

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Interrupt number	Interrupt type
32	Right stop switch 2
33	Left stop switch 3
34	Right stop switch 3
35	Left stop switch 4
36	Right stop switch 4
37	Left stop switch 5
38	Right stop switch 5
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

Table 12: Interrupt Vectors

3.6.6.2 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 (i.e. accumulator register, X register, flags) will be saved automatically.
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 (RETI command), the context of the normal program will automatically be restored and the execution of the normal program will be continued.

3.6.6.3 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 (chapter 3.7.9) for further information about that.

3.6.6.4 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.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

- An interrupt handling routine must always end with a RETI command.
- Do not allow the normal program flow to run into an interrupt handling routine.

The following example shows 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 the example above, the interrupt numbers are being 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 as follows:

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

3.6.7 New TMCL Commands

In order to make several operations easier, the following new commands have been introduced from firmware version 1.45 on. Using these new commands many taks can be programmed in an easier way. This can save some code, thus making a TMCL program shorter, faster and easier to understand.

Please note that these commands are not available on TMCM-6110 modules with firmware versions before 1.45. So please make sure that at least firmware version 1.45 is installed before using them.

New TMCL Commands		
Mnemonic	Command number	Meaning
CALCVV	40	Calculate using two user variables
CALCVA	41	Calculate using a user variable and the accumulator

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Mnemonic	Command number	Meaning
CALCAV	42	Calculate using the accumulator and a user variable
CALCVX	43	Calculate using a user variable and the X register
CALCXV	44	Calculate using the X register and a user variable
CALCV	45	Calculate using a user variable and a direct value
MVPA	46	Move to position specified by accumulator
MVPXA	47	Move to position specified by accumulator and X register
RST	48	Restart the program
DJNZ	49	Decrement and jump if not zero
CALL	80	Conditional subroutine call
ROLA	50	Rotate left using the accumulator
RORA	51	Rotate right using the accumulator
ROLXA	52	Rotate left using accumulator and X register
RORXA	53	Rotate right using accumulator and X register
MSTX	54	Stop motor specified by X register
SAPX	16	Set axis parameter, axis specified by X register
GAPX	17	Get axis parameter, axis specified by X register
AAPX	18	Copy accumulator to an axis parameter, axis specified by X register
SIV	55	Set indexed variable
GIV	56	Get indexed variable
AIV	57	Accu to indexed variable

Table 13: New TMCL Commands

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7 Detailed TMCL Command Descriptions

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

3.7.1 ROR (Rotate Right)

The motor is instructed to rotate with a specified velocity in right direction (increasing the position counter). The velocity is given in internal units ([int]) of the TMC429 motion controller used on the TMCM-6110 module. Please see also section 6.1.

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

Related commands: ROL, MST, SAP, GAP.

Mnemonic: ROR <axis>, <velocity>

Binary Representation			
Instruction	Type	Motor/Bank	Value
2	0	0…5	-2047…2047

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Rotate right motor 0, velocity 1000.
Mnemonic: ROR 0, 1000.

Binary Form of ROR 0, 1000	
Field	Value
Target address	01_h
Instruction number	01_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	03_h
Value (Byte 0)	$E8_h$
Checksum	ED_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.2 ROL (Rotate Left)

The motor is instructed to rotate with a specified velocity in left direction (decreasing the position counter). The velocity is given in internal units ([int]) of the TMC429 motion controller used on the TMCM-6110 module. Please see also section 6.1.

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

Related commands: ROR, MST, SAP, GAP.

Mnemonic: ROL <axis>, <velocity>

Binary Representation			
Instruction	Type	Motor/Bank	Value
2	0	0...5	-2047...2047

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Rotate left motor 0, velocity 1000.
Mnemonic: ROL 0, 1000.

Binary Form of ROL 0, 1000	
Field	Value
Target address	01_h
Instruction number	02_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	03_h
Value (Byte 0)	$E8_h$
Checksum	EE_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.3 MST (Motor Stop)

The MST command stops the motor using a soft stop.

Internal function: The velocity mode is selected. Then, the target velocity (axis parameter #2) is set to zero.

Related commands: ROR, ROL, SAP, GAP.

Mnemonic: MST <axis>

Binary Representation			
Instruction	Type	Motor/Bank	Value
3	0	0...5	0

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Stop motor 0.
Mnemonic: MST 0.

Binary Form of MST 0	
Field	Value
Target address	01_h
Instruction number	03_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	04_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.4 MVP (Move to Position)

With this command the motor will be instructed to move to a specified relative or absolute position. 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 as well as other ramp parameters are defined by the appropriate axis parameters. For a list of these parameters please refer to section 4.
The range of the MVP command is 32 bit signed (-2147483648...2147483647). Positioning can be interrupted using MST, ROL or ROR commands.

Three operation types are available:

- Moving to an absolute position in the range from -2147483648...2147483647 ($-2^{31}...2^{31}-1$).
- Starting a relative movement by means of an offset to the actual position. In this case, the new resulting position value must not exceed the above mentioned limits, too.
- Moving the motor to a (previously stored) coordinate (refer to SCO for details).

Note The distance between the actual position and the new position must not be more than 2147483647 ($2^{31}-1$) microsteps . Otherwise the motor will run in the opposite direction in order to take the shorter distance (caused by 32 bit overflow).

Internal function: Position mode is selected and the new position value is transferred to axis parameter #0 (target position).

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

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

Binary Representation			
Instruction	Type	Motor/Bank	Value
4	0 – ABS – absolute	0...5	<position>
	1 – REL – relative	0...5	<offset>
	2 – COORD – coordinate	0...255	<coordinate number (0..20)>

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Move motor 0 to position 90000.
Mnemonic: MVP ABS, 0, 90000

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of MVP ABS, 0, 90000	
Field	Value
Target address	01_h
Instruction number	04_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	01_h
Value (Byte 1)	$5F_h$
Value (Byte 0)	90_h
Checksum	$F5_h$

Example
Move motor 0 from current position 10000 steps backward.
Mnemonic: MVP REL, 0, -10000

Binary Form of MVP REL, 0, -10000	
Field	Value
Target address	01_h
Instruction number	04_h
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	FF_h
Value (Byte 2)	FF_h
Value (Byte 1)	$D8_h$
Value (Byte 0)	$F0_h$
Checksum	CC_h

Example
Move motor 0 to stored coordinate #8.

Mnemonic: MVP COORD, 0, 8

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of MVP COORD, 0, 8	
Field	Value
Target address	01_h
Instruction number	04_h
Type	02_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	08_h
Checksum	$0F_h$

Note

Before moving to a stored coordinate, the coordinate has to be set using an SCO, CCO or ACO command.

Info

The following options are available for the MVP COORD command:

- Moving only one motor: Set the <motor> parameter to the motor number (0...5).
- Moving multiple motors without interpolation: Set bit 7 of the <motor> parameter. Now the bits 0...5 of the <motor> parameter define which motors are to be started. Each of these bits stands for one motor. The number of motors that can be started simulataneously without interpolation is not limited.
- Moving multiple motors using interpolation: Set bit 6 of the <motor> parameter. Now the bits 0...5 of the <motor> parameter define which motors are to be moved using interpolation. Each of these bits stands for one motor. Interpolated movements are only possible for two or three motors at a time. If more than three bits are set then only the first three of these motors will be used. A simple linear interpolation algorithm is being used: the velocities and accelerations/decelerations of all motors will be automatically adjusted in such a way that all participating motors will reach their target positions at the same time.

Examples:

- Move motors #0, #1 and #2 to coordinate #2 using interpolation: MVP COORD, $47, 2.
- Move motors #0 and #2 to coordinate #5 without interpolation: MVP COORD, $85, 5.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.5 SAP (Set Axis Parameter)

With this command most of the motion control parameters of the module can be specified. The settings will be stored in SRAM and therefore are volatile. That is, information will be lost after power off.

Info	For a table with parameters and values which can be used together with this command please refer to section 4.

Internal function: The specified value is written to the axis parameter specified by the parameter number.
Related commands: GAP, AAP.

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

Binary representation

Binary Representation			
Instruction	Type	Motor/Bank	Value
5	see chapter 4	0...5	<value>

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example Set the maximum positioning speed for motor 0 to 1000.
Mnemonic: SAP 4, 0, 1000.

Binary Form of SAP 4, 0, 1000	
Field	Value
Target address	01_h
Instruction number	05_h
Type	04_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	03_h
Value (Byte 0)	$E8_h$
Checksum	$F5_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.6 GAP (Get Axis Parameter)

Most motion / driver related parameters of the TMCM-6110 can be adjusted using e.g. the SAP command. With the GAP 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 conditional jumps). In direct mode the value read is only output in the value field of the reply, without affecting the accumulator.

Info	For a table with parameters and values that can be used together with this command please refer to section 4.

Internal function: The specified value gets copied to the accumulator.
Related commands: SAP, AAP.

Mnemonic: GAP <parameter number>, <axis>

Binary Representation			
Instruction	Type	Motor/Bank	Value
6	see chapter 4	0…5	<value>

Reply in Direct Mode	
Status	Value
100 - OK	value read by this command

Example
Get the actual position of motor 0.
Mnemonic: GAP 1, 0.

Binary Form of GAP 1, 0	
Field	Value
Target address	01_h
Instruction number	06_h
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	08_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.7 STAP (Store Axis Parameter)

This command is used to store TMCL axis parameters permanently in the EEPROM of the module. This command is mainly needed to store the default configuration of the module. The contents of the user variables can either be automatically or manually restored at power on.

Info	For a table with parameters and values which can be used together with this command please refer to section 4.

Internal function: The axis parameter specified by the type and bank number will be stored in the EEPROM.

Related commands: SAP, AAP, GAP, RSAP.

Mnemonic: STAP <parameter number>, <bank>

Binary Representation			
Instruction	Type	Motor/Bank	Value
7	see chapter 4	0...5	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	0 (don't care)

Example
Store axis parameter #6.
Mnemonic: STAP 7, 6.

Binary Form of STAP 6, 12	
Field	Value
Target address	01_h
Instruction number	07_h
Type	06_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$0E_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.8 RSAP (Restore Axis Parameter)

With this command the contents of an axis parameter can be restored from the EEPROM. By default, all axis parameters are automatically restored after power up. An axis parameter that has been changed before can be reset to the stored value by this instruction.

Info	For a table with parameters and values which can be used together with this command please refer to section 4.

Internal function: The axis parameter specified by the type and bank number will be restored from the EEPROM.

Related commands: SAP, AAP, GAP, RSAP.

Mnemonic: RSAP <parameter number>, <bank>

Binary Representation			
Instruction	Type	Motor/Bank	Value
8	see chapter 4	0...5	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	0 (don't care)

Example
Restore axis parameter #6.
Mnemonic: RSAP 8, 6.

Binary Form of RSAP 8, 6	
Field	Value
Target address	01_h
Instruction number	08_h
Type	06_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$0A_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.9 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 is used for global parameters, and bank 2 is used for user variables. Bank 3 is used for interrupt configuration.
All module settings in bank 0 will automatically be stored in non-volatile memory (EEPROM).

Info	For a table with parameters and values which can be used together with this command please refer to section 5.

Internal function: The specified value will be copied to the global parameter specified by the type and bank number. Most parameters of bank 0 will automatically be stored in non-volatile memory.
Related commands: GGP, AGP.

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

Binary Representation			
Instruction	Type	Motor/Bank	Value
9	see chapter 5	0/2/3	<value>

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Set the serial address of the device to 3.
Mnemonic: SGP 66, 0, 3.

Binary Form of SGP 66, 0, 3	
Field	Value
Target address	01_h
Instruction number	09_h
Type	42_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	03_h
Checksum	$4F_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.10 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 is used for global parameters, and bank 2 is used for user variables. Bank 3 is used for interrupt configuration.

Info	For a table with parameters and values which can be used together with this command please refer to section 5.

Internal function: The global parameter specified by the type and bank number will be copied to the accumulator register.
Related commands: SGP, AGP.

Mnemonic: GGP <parameter number>, <bank>

Binary Representation			
Instruction	Type	Motor/Bank	Value
10	see chapter 5	0/2/3	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	value read by this command

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

Binary Form of GGP 66, 0	
Field	Value
Target address	01_h
Instruction number	$0A_h$
Type	42_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$4D_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.11 STGP (Store Global Parameter)

This command is used to store TMCL global parameters permanently in the EEPROM of the module. This command is mainly needed to store the TMCL user variables (located in bank 2) in the EEPROM of the module, as most other global parameters (located in bank 0) are stored automatically when being modified. The contents of the user variables can either be automatically or manually restored at power on.

Info For a table with parameters and values which can be used together with this command please refer to section 5.3.

Internal function: The global parameter specified by the type and bank number will be stored in the EEPROM.

Related commands: SGP, AGP, GGP, RSGP.

Mnemonic: STGP <parameter number>, <bank>

Binary Representation			
Instruction	Type	Motor/Bank	Value
11	see chapter 5.3	2	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	0 (don't care)

Example
Store user variable #42.
Mnemonic: STGP 42, 2.

Binary Form of STGP 42, 2	
Field	Value
Target address	01_h
Instruction number	$0B_h$
Type	$2A_h$
Motor/Bank	02_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	38_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.12 RSGP (Restore Global Parameter)

With this command the contents of a TMCL user variable can be restored from the EEPROM. By default, all user variables are automatically restored after power up. A user variable that has been changed before can be reset to the stored value by this instruction.

Info For a table with parameters and values which can be used together with this command please refer to section 5.3.

Internal function: The global parameter specified by the type and bank number will be restored from the EEPROM.

Related commands: SGP, AGP, GGP, STGP.

Mnemonic: RSGP <parameter number>, <bank>

Binary Representation			
Instruction	Type	Motor/Bank	Value
12	see chapter 5.3	2	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	0 (don't care)

Example
Restore user variable #42.
Mnemonic: RSGP 42, 2.

Binary Form of RSGP 42, 2	
Field	Value
Target address	01_h
Instruction number	$0C_h$
Type	$2A_h$
Motor/Bank	02_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	39_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.13 RFS (Reference Search)

The TMCM-6110 has a built-in reference search algorithm. The reference search algorithm provides different refrence search modes. This command starts or stops the built-in reference search algorithm. The status of the reference search can also be queried to see if it already has finished. (In a TMCL program it mostly is better to use the WAIT RFS 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 (please see chapter 4).

Internal function: The internal reference search state machine is started or stoped, or its state is queried.

Related commands: SAP, GAP, WAIT.

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

Binary Representation			
Instruction	Type	Motor/Bank	Value
13	0 START — start reference search 1 STOP — stop reference search 2 STATUS — get status	0...5	0 (don't care)

Reply in Direct Mode (RFS START or RFS STOP)	
Status	Value
100 - OK	0 (don't care)

Reply in Direct Mode (RFS STATUS)		
Status	Value	
100 - OK	0	no ref. search active
	other values	reference search active

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

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of RFS START	
Field	Value
Target address	01_h
Instruction number	$0D_h$
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$0E_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.14 SIO (Set Output)

This command sets the states of the general purpose digital outputs.

Internal function: The state of the output line specified by the type parameter is set according to the value passed to this command.

Related commands: GIO.

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

Binary Representation			
Instruction	Type	Motor/Bank	Value
14	<port number>	<bank number> (2)	0/1

Reply in Direct Mode	
Status	Value
100 - OK	0 (don't care)

Example
Set output 0 (bank 2) to high.
Mnemonic: SIO 0, 2, 1.

Binary Form of SIO 0, 2, 1	
Field	Value
Target address	01_h
Instruction number	$0E_h$
Type	00_h
Motor/Bank	02_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	01_h
Checksum	12_h

Bank 2 – Digital Outputs
The following output lines can be set by the SIO commands) using bank 2.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Digital Outputs in Bank 2		
Port	Command	Range
OUT0	SIO 0, 2, <value>	0/1
OUT1	SIO 1, 2, <value>	0/1
OUT2	SIO 2, 2, <value>	0/1
OUT3	SIO 3, 2, <value>	0/1
OUT4	SIO 4, 2, <value>	0/1
OUT5	SIO 5, 2, <value>	0/1
OUT6	SIO 6, 2, <value>	0/1
OUT7	SIO 7, 2, <value>	0/1

Special case: SIO 255, 2, <x> can be used to change all general purpose digital output lines simulaneously. The value <x> will then be interpreted as a bit vector where each of the lower eight bits represents one of the digital outputs. So the range for <x> is 0…255. The value <x> can also be -1. In this case, the value will be taken from the accumulator register. The following program can be used to copy the states of the input lines to the output lines:

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

Pull-up Resistor Control
The SIO command can also be used to program the pull-up resistors of the STOP_L and STOP_R inputs. The pull-up resistors can be switched in two groups: those for motors 0, 1 and 2 and those for motors 3, 4 and 5. Use SIO 0, 0, <x> to switch the pull-up resistors. The value <x> will be interpreted as a bit vector: bit 0 controls the pull-up resistors for motors 0, 1 and 2 and bit 1 controls the pull-up resistors for motors 3, 4 and 5.
So the following values can be set:

Pull-up Resistor Control	
Command	Pull-up resistors
SIO 0, 0, 0	All off
SIO 0, 0, 1	Motor 0, 1, 2: on Motor 3, 4, 5: off
SIO 0, 0, 2	Motor 0, 1, 2: off Motor 3, 4, 5: on
SIO 0, 0, 3	All on

All pull-up resistors are on by default.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.15 GIO (Get Input)

With this command the status of the available general purpose outputs of the module can be read. The function reads a digital or an analog input port. Digital lines will read as 0 or 1, while the ADC channels deliver their 12 bit result in the range of 0...4095. In standalone mode the requested value is copied to the accumulator register for further processing purposes such as conditional jumps. In direct mode the value is only output in the value field of the reply, without affecting the accumulator. The actual status of a digital output line can also be read.

Internal function: The state of the i/o line specified by the type parameter and the bank parameter is read.

Related commands: SIO.

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

Binary Representation			
Instruction	Type	Motor/Bank	Value
15	<port number>	<bank number> (0/1/2)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	status of the port

Example
Get the value of ADC channel 0.
Mnemonic: GIO 0, 1.

Binary Form of GIO 0, 1	
Field	Value
Target address	01_h
Instruction number	$0F_h$
Type	00_h
Motor/Bank	01_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	11_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Reply (Status=no error, Value=302)	
Field	Value
Host address	02_h
Target address	01_h
Status	64_h
Instruction	$0F_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	01_h
Value (Byte 0)	$2E_h$
Checksum	$A5_h$

Bank 0 – Digital Inputs
The analog input lines can be read as digital or analog inputs at the same time. The digital input states can be accessed in bank 0.

Digital Inputs in Bank 0		
Port	Command	Range
AIN0	GIO 0, 0	0/1
IN1	GIO 1, 0	0/1
IN2	GIO 2, 0	0/1
IN3	GIO 3, 0	0/1
AIN4	GIO 4, 0	0/1
IN5	GIO 5, 0	0/1
IN6	GIO 6, 0	0/1
IN7	GIO 7, 0	0/1

Special case: GIO 255, 0 reads all general purpose inputs simulataneously and puts the result into the lower eight bits of the accumulator register.

Bank 1 – Analog Inputs
The analog input lines can be read back as digital or analog inputs at the same time. The analog values can be accessed in bank 1.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Analog Inputs in Bank 1		
Port	Command	Range
AIN0	GIO 0, 1	0…4095
AIN4	GIO 4, 1	0…4095
Supply Voltage	GIO 8, 1	[1/10V]

Bank 2 – States of the Digital Outputs
The states of the output lines (that have been set by SIO commands) can be read back using bank 2.

Digital Outputs in Bank 2		
Port	Command	Range
OUT0	GIO 0, 2	0/1
OUT1	GIO 1, 2	0/1
OUT2	GIO 2, 2	0/1
OUT3	GIO 3, 2	0/1
OUT4	GIO 4, 2	0/1
OUT5	GIO 5, 2	0/1
OUT6	GIO 6, 2	0/1
OUT7	GIO 7, 2	0/1

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.16 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. *This command is mainly intended for use in standalone mode.*

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

Mnemonic: CALC <operation>, <operand>

Binary representation

Binary Representation			
Instruction	Type	Motor/Bank	Value
19	0 ADD – add to accumulator 1 SUB – subtract from accumulator 2 MUL – multiply accumulator by 3 DIV – divide accumulator by 4 MOD – modulo divide accumulator by 5 AND – logical and accumulator with 6 OR – logical or accumulator with 7 XOR – logical exor accumulator with 8 NOT – logical invert accumulator 9 LOAD – load operand into accumulator	0 (don't care)	<operand>

Reply in Direct Mode	
Status	Value
100 - OK	the operand (don't care)

Example
Multiply accumulator by -5000.
Mnemonic: CALC MUL, -5000

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of CALC MUL, -5000	
Field	Value
Target address	01_h
Instruction number	13_h
Type	02_h
Motor/Bank	00_h
Value (Byte 3)	FF_h
Value (Byte 2)	FF_h
Value (Byte 1)	EC_h
Value (Byte 0)	78_h
Checksum	78_h

Reply (Status=no error, value=-5000:	
Field	Value
Host address	02_h
Target address	01_h
Status	64_h
Instruction	13_h
Value (Byte 3)	FF_h
Value (Byte 2)	FF_h
Value (Byte 1)	EC_h
Value (Byte 0)	78_h
Checksum	DC_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.17 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 accumulator register is compared with the sepcified value. The internal arithmetic status flags are set according to the result of the comparison. These can then control e.g. a conditional jump.
Related commands: JC, GAP, GGP, GIO, CALC, CALCX.

Mnemonic: COMP <operand>

Binary Representation			
Instruction	Type	Motor/Bank	Value
20	0 (don't care)	0 (don't care)	<operand>

Example
Jump to the address given by the label when the position of motor #0 is greater than or equal to 1000.

```
GAP 1, 0 //get actual position of motor 0
COMP 1000   //compare actual value with 1000
JC GE, Label   //jump to Lable if greter or equal to 1000
```

Binary Form of COMP 1000	
Field	Value
Target address	01_h
Instruction number	14_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	03_h
Value (Byte 0)	$E8_h$
Checksum	00_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.18 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 command is intended for standalone operation only.*

Internal function: The TMCL program counter is set to the value passed to this command if the status flags are in the appropriate states.

Related commands: JA, COMP, WAIT, CLE.

Mnemonic: JC <condition>, <label>

Binary Representation			
Instruction	Type	Motor/Bank	Value
21	0 ZE - zero 1 NZ - not zero 2 EQ - equal 3 NE - not equal 4 GT - greater 5 GE - greater/equal 6 LT - lower 7 LE - lower/equal 8 ETO - time out error 9 EAL - external alarm 10 EDV - deviation error 11 EPO - position error	0 (don't care)	<jump address>

Example
Jump to the address given by the label when the position of motor #0 is greater than or equal to 1000.

```
GAP 1, 0  //get actual position of motor 0
COMP 1000   //compare actual value with 1000
JC GE, Label  //jump to Lable if greter or equal to 1000
...
Label: ROL 0, 1000
```

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary form of JC GE, Label assuming Label at address 10	
Field	Value
Target address	01_h
Instruction number	15_h
Type	05_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	$0A_h$
Checksum	25_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.19 JA (Jump always)

Jump to a fixed address in the TMCL program memory. *This command is intended for standalone operation only.*

Internal function: The TMCL program counter is set to the value passed to this command.

Related commands: JC, WAIT, CSUB.

Mnemonic: JA <label>

Binary Representation			
Instruction	Type	Motor/Bank	Value
22	0 (don't care)	0 (don't care)	<jump address>

Example
An infinite loop in TMCL:

```
Loop:
    MVP ABS, 0, 51200
    WAIT POS, 0, 0
    MVP ABS, 0, 0
    WAIT POS, 0, 0
    JA Loop
```

Binary form of the JA Loop command when the label Loop is at address 10:

Binary Form of JA Loop (assuming Loop at address 10)	
Field	Value
Target address	01_h
Instruction number	16_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	$0A_h$
Checksum	21_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.20 CSUB (Call Subroutine)

This function calls a subroutine in the TMCL program memory. *It 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	Type	Motor/Bank	Value
23	0 (don't care)	0 (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
    CSUB SubW   //Save program counter and jump to label SubW
    JA Loop

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

Binary form of CSUB SubW (assuming SubW at address 100)	
Field	Value
Target address	01_h
Instruction number	17_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	64_h
Checksum	$7C_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.21 RSUB (Return from Subroutine)

Return from a subroutine to the command after the CSUB command. *This command is intended for use in standalone mode only.*

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

Related commands: CSUB.

Mnemonic: RSUB

Binary Representation			
Instruction	Type	Motor/Bank	Value
24	0 (don't care)	0 (don't care)	0 (don't care)

Example
Please see the CSUB example (section 3.7.20).
Binary form:

Binary Form of RSUB	
Field	Value
Target address	01_h
Instruction number	18_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	19_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.22 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.

Special case for the <ticks> parameter: When this parameter is set to -1 the contents of the accumulator register will be taken for this value. So for example WAIT TICKS, 0, -1 will wait as long as specified by the value store in the accumulator. *The accumulator must not contain a negative value when using this option.*

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 will be held at the address of this WAIT command until the condition is met or the timeout has expired.

Related commands: JC, CLE.

Mnemonic: WAIT <condition>, <motor number>, <ticks>

Binary Representation			
Instruction	Type	Motor/Bank	Value
27	0 TICKS – timer ticks	0 (don't care)	<no. of ticks to wait[1]>
	1 POS – target position reached	<motor number>	<no. of ticks for timeout[1]> 0 for no timeout
	2 REFSW – reference switch	<motor number>	<no. of ticks for timeout[1]> 0 for no timeout
	3 LIMSW – limit switch	<motor number>	<no. of ticks for timeout[1]> 0 for no timeout
	4 RFS – reference search completed	<motor number>	<no. of ticks for timeout[1]> 0 for no timeout

Example

[1] one tick is 10 milliseconds

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

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

Binary Form of WAIT POS, 0, 0	
Field	Value
Target address	01_h
Instruction number	$1B_h$
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$1D_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.23 STOP (Stop TMCL Program Execution – End of TMCL Program)

This command stops the execution of a TMCL program. *It is intended for use in standalone operation only.*

Internal function: Execution of a TMCL program in standalone mode will be stopped.

Related commands: none.

Mnemonic: STOP

Binary Representation			
Instruction	Type	Motor/Bank	Value
28	0 (don't care)	0 (don't care)	0 (don't care)

Example
Mnemonic: STOP

Binary Form of STOP	
Field	Value
Target address	01_h
Instruction number	$1C_h$
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$1D_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.24 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).

Note Coordinate #0 is always stored in RAM only.

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

Related commands: GCO, CCO, ACO, MVP COORD.

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

Binary Representation			
Instruction	Type	Motor/Bank	Value
30	<coordinate number> 0...20	<motor number> 0...5	<position> $-2^{31} \ldots 2^{31}-1$

Example
Set coordinate #1 of motor #0 to 1000.
Mnemonic: SCO 1, 0, 1000

Binary Form of SCO 1, 0, 1000	
Field	Value
Target address	01_h
Instruction number	$1E_h$
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	03_h
Value (Byte 0)	$E8_h$
Checksum	$0B_h$

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.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.25 GCO (Get Coordinate)

Using this command previously stored coordinate can be read back. In standalone mode the requested value is copied to the accumulator register for further processing purposes such as conditional 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).

Note Coordinate #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, ACO, MVP COORD.

Mnemonic: GCO <coordinate number>, <motor number>

Binary Representation			
Instruction	Type	Motor/Bank	Value
31	<coordinate number> 0...20	<motor number> 0...5	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	value read by this command

Example
Get coordinate #1 of motor #0.
Mnemonic: GCO 1, 0

Binary Form of GCO 1, 0	
Field	Value
Target address	01_h
Instruction number	$1F_h$
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	21_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

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.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.26 CCO (Capture Coordinate)

This command copies the actual position of the axis 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	Coordinate #0 is always stored in RAM only.

Internal function: the actual position of the selected motor is copied to selected coordinate array entry.

Related commands: SCO, GCO, ACO, MVP COORD.

Mnemonic: CCO <coordinate number>, <motor number>

Binary Representation			
Instruction	Type	Motor/Bank	Value
32	<coordinate number> 0…20	<motor number> 0…5	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	value read by this command

Example
Store current position of motor #0 to coordinate array entry #3.

Mnemonic: CCO 3, 0

Binary Form of CCO 3, 0	
Field	Value
Target address	01_h
Instruction number	20_h
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	22_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.27 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).

Note Coordinate #0 is always stored in RAM only.

Internal function: the actual position of the selected motor is copied to selected coordinate array entry.

Related commands: SCO, GCO, CO, MVP COORD.

Mnemonic: ACO <coordinate number>, <motor number>

Binary Representation			
Instruction	Type	Motor/Bank	Value
39	<coordinate number> 0...20	<motor number> 0...5	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Copy the actual value of the accumulator to coordinate #1 of motor #0.

Mnemonic: ACO 1, 0

Binary Form of ACO 1, 0	
Field	Value
Target address	01_h
Instruction number	27_h
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	29_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.28 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. *This command is mainly intended for use in standalone mode.*

Related commands: CALC, COMP, JC, AAP, AGP, GAP, GGP, GIO.

Mnemonic: CALCX <operation>

Binary Representation			
Instruction	Type	Motor/Bank	Value
33	0 ADD – add X register to accumulator 1 SUB – subtract X register from accumulator 2 MUL – multiply accumulator by X register 3 DIV – divide accumulator by X register 4 MOD – modulo divide accumulator by X register 5 AND – logical and accumulator with X register 6 OR – logical or accumulator with X register 7 XOR – logical exor accumulator with X register 8 NOT – logical invert X register 9 LOAD – copy accumulator to X register 10 SWAP – swap accumulator and X register	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Multiply accumulator and X register.
Mnemonic: CALCX MUL

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of CALCX MUL	
Field	Value
Target address	01_h
Instruction number	21_h
Type	02_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	24_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.29 AAP (Accu 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. *This command is mainly intended for use in standalone mode.*

Info For a table with parameters and values which can be used together with this command please refer to section 4.

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

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

Binary Representation			
Instruction	Type	Motor/Bank	Value
34	see chapter 4	0...5	<value>

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Position motor #0 by a potentiometer connected to analog input #0:

```
Start:
    GIO 0,1       //get value of analog 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 Form of AAP 0, 0	
Field	Value
Target address	01_h
Instruction number	22_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	23_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.30 AGP (Accu 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. *This command is mainly intended for use in standalone mode.*

Info	For an overview of parameter and bank indices that can be used with this command please see section 5.

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

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

Binary Representation			
Instruction	Type	Motor/Bank	Value
35	<parameter number>	0/2/3 <bank number>	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Copy accumulator to user variable #42:
Mnemonic: AGP 42, 2

Binary Form of AGP 42, 2	
Field	Value
Target address	01_h
Instruction number	23_h
Type	$2A_h$
Motor/Bank	02_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	50_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.31 CLE (Clear Error Flags)

This command clears the internal error flags. It is mainly intended for use in standalone mode.
The following error flags can be cleared by this command (determined by the <flag> parameter):

- ALL: clear all error flags.
- ETO: clear the timeout flag.
- EAL: clear the external alarm flag.
- EDV: clear the deviation flag.
- EPO: clear the position error flag.

Related commands: JC, WAIT.

Mnemonic: CLE <flags>

Binary Representation			
Instruction	Type	Motor/Bank	Value
36	0 ALL – all flags 1 – (ETO) timeout flag 2 – (EAL) alarm flag 3 – (EDV) deviation flag 4 – (EPO) position flag 5 – (ESD) shutdown flag	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Reset the timeout flag.
Mnemonic: CLE ETO

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of CLE ETO	
Field	Value
Target address	01_h
Instruction number	24_h
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	26_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.32 EI (Enable Interrupt)

The EI command enables an interrupt. It needs the interrupt number as parameter. Interrupt number 255 globally enables interrupt processing. *This command is mainly intended for use in standalone mode.*

Info	Please see table 12 for a list of interrupts that can be used on the TMCM-6110 module.

Related commands: DI, VECT, RETI.

Mnemonic: EI <interrupt number>

Binary Representation			
Instruction	Type	Motor/Bank	Value
25	<interrupt number>	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Globally enable interrupt processing:
Mnemonic: EI 255

Binary form of EI 255	
Field	Value
Target address	01_h
Instruction number	19_h
Type	FF_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	19_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.33 DI (Disable Interrupt)

The DI command disables an interrupt. It needs the interrupt number as parameter. Interrupt number 255 globally disables interrupt processing. *This command is mainly intended for use in standalone mode.*

Info	Please see table 12 for a list of interrupts that can be used on the TMCM-6110 module.

Related commands: EI, VECT, RETI.

Mnemonic: DI <interrupt number>

Binary Representation			
Instruction	Type	Motor/Bank	Value
26	<interrupt number>	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Globally disable interrupt processing:
Mnemonic: DI 255

Binary Form of DI 255	
Field	Value
Target address	01_h
Instruction number	$1A_h$
Type	FF_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$1A_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.34 VECT (Define Interrupt Vector)

The VECT command defines an interrupt vector. It takes an interrupt number and a label (just like with JA, JC and CSUB commands) as parameters. The label must be the entry point of the interrupt handling routine for this interrupts. Interrupt vectors can also be re-defined. *This command is intended for use in standalone mode only.*

Info	Please see table 12 for a list of interrupts that can be used on the TMCM-6110 module.

Related commands: EI, DI, RETI.

Mnemonic: VECT <interrupt number>, <label>

Binary Representation			
Instruction	Type	Motor/Bank	Value
37	<interrupt number>	0 (don't care)	<label>

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Define interrupt vector for timer #0 interrupt:

```
    VECT 0, Timer0Irq
    ...
Loop:
    ...
    JA Loop
    ...
Timer0Irq:
    SIO 0, 2, 1
    RETI
```

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary form of VECT (assuming label is at 50)	
Field	Value
Target address	01_h
Instruction number	25_h
Type	FF_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	32_h
Checksum	58_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.35 RETI (Return from Interrupt)

This command terminates an interrupt handling routine. Normal program flow will be continued then. *This command is intended for use in standalone mode only.*

An interrupt routine must always end with a RETI command. Do not allow the normal program flow to run into an interrupt routine.

Internal function: The saved registers (accumulator, X registers, flags and program counter) are copied back so that normal program flow will continue.

Related commands: EI, DI, VECT.

Mnemonic: RETI

Binary Representation			
Instruction	Type	Motor/Bank	Value
38	<interrupt number>	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Return from an interrup handling routine.
Mnemonic: RETI

Binary Form of RETI	
Field	Value
Target address	01_h
Instruction number	26_h
Type	FF_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	27_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.36 CALCVV (Calculate using two User Variables)

The CALCVV instruction directly uses the contents of two user variables for an arithmetic operation, storing the result in the first user variable. This eliminates the need for using the accumulator register and/or X register for such purposes. The parameters of this command are the arithmetic function, the index of the first user variable (0...255) and the index of the second user variable (0...255). *This command is mainly intended for use in standalone mode.*

Related commands: CALCVA, CALCAV, CALCVX, CALCXV, CALCV.

Mnemonic: CALCVV <operation>, <var1>, <var2>

Binary representation

Binary Representation			
Instruction	Type	Motor/Bank	Value
40	0 ADD – add <var2> to <var1> 1 SUB – subtract <var2> from <var1> 2 MUL – multiply <var2> with <var1> 3 DIV – divide <var2> by <var1> 4 MOD – modulo divide <var2> by <var1> 5 AND – logical and <var2> with <var1> 6 OR – logical or <var2> with <var1> 7 XOR – logical exor <var2> with <var1> 8 NOT – copy logical inverted <var2> to <var1> 9 LOAD – copy <var2> to <var1> 10 SWAP – swap contents of <var1> and <var2> 11 COMP – compare <var1> with <var2>	0 <var1> (0...255)	<var2> (0...255)

Reply in Direct Mode	
Status	Value
100 - OK	the operand (don't care)

Example
Subtract user variable #42 from user variable #65.
Mnemonic: CALCVV SUB, 65, 42

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of CALCVV SUB, 65, 42	
Field	Value
Target address	01_h
Instruction number	28_h
Type	01_h
Motor/Bank	41_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	$2A_h$
Checksum	95_h

Reply (Status=no error, value=0:	
Field	Value
Host address	02_h
Target address	01_h
Status	64_h
Instruction	28_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$8F_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.37 CALCVA (Calculate using a User Variable and the Accumulator Register)

The CALCVA instruction directly modifies a user variable using an arithmetical operation and the contents of the accumulator register. The parameters of this command are the arithmetic function and the index of a user variable (0...255). *This command is mainly intended for use in standalone mode.*

Related commands: CALCV, CALCAV, CALCVX, CALCXV, CALCVV.

Mnemonic: CALCVA <operation>, <var>

Binary representation

Binary Representation			
Instruction	Type	Motor/Bank	Value
41	0 ADD – add accumulator to <var> 1 SUB – subtract accumulator from <var> 2 MUL – multiply <var> with accumulator 3 DIV – divide <var> by accumulator 4 MOD – modulo divide <var> by accumulator 5 AND – logical and <var> with accumulator 6 OR – logical or <var> with accumulator 7 XOR – logical exor <var> with accumulator 8 NOT – copy logical inverted accumulator to <var> 9 LOAD – copy accumulator to <var> 10 SWAP – swap contents of <var> and accumulator 11 COMP – compare <var> with accumulator	0 <var> (0...255)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	the operand (don't care)

Example
Subtract accumulator from user variable #27.
Mnemonic: CALCVA SUB, 27

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of CALCVA SUB, 27	
Field	Value
Target address	01_h
Instruction number	29_h
Type	01_h
Motor/Bank	$1B_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	46_h

Reply (Status=no error, value=0:	
Field	Value
Host address	02_h
Target address	01_h
Status	64_h
Instruction	29_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	90_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.38 CALCAV (Calculate using the Accumulator Register and a User Variable)

The CALCAV instruction modifies the accumulator register using an arithmetical operation and the contents of a user variable. The parameters of this command are the arithmetic function and the index of a user variable (0...255). *This command is mainly intended for use in standalone mode.*

Related commands: CALCV, CALCAV, CALCVX, CALCXV, CALCVV.

Mnemonic: CALCAV <operation>, <var>

Binary representation

Binary Representation			
Instruction	Type	Motor/Bank	Value
42	0 ADD – add <var> to accumulator 1 SUB – subtract <var> from accumulator 2 MUL – multiply accumulator with <var> 3 DIV – divide accumulator by <var> 4 MOD – modulo divide accumulator by <var> 5 AND – logical and accumulator with <var> 6 OR – logical or accumulator with <var> 7 XOR – logical exor accumulator with <var> 8 NOT – copy logical inverted <var> to accumulator 9 LOAD – copy <var> to accumulator 10 SWAP – swap contents of <var> and accumulator 11 COMP – compare accumulator with <var>	0 <var> (0...255)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	the operand (don't care)

Example
Subtract user variable #27 from accumulator.
Mnemonic: CALCXV SUB, 27

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of CALCXV SUB, 27	
Field	Value
Target address	01_h
Instruction number	$2A_h$
Type	01_h
Motor/Bank	$1B_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	47_h

Reply (Status=no error, value=0:	
Field	Value
Host address	02_h
Target address	01_h
Status	64_h
Instruction	$2A_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	91_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.39 CALCVX (Calculate using a User Variable and the X Register)

The CALCVX instruction directly modifies a user variable using an arithmetical operation and the contents of the X register. The parameters of this command are the arithmetic function and the index of a user variable (0...255). *This command is mainly intended for use in standalone mode.*

Related commands: CALCV, CALCAV, CALCVA, CALCXV, CALCVV.

Mnemonic: CALCVX <operation>, <var>

Binary representation

Binary Representation			
Instruction	Type	Motor/Bank	Value
43	0 ADD – add X register to <var> 1 SUB – subtract X register from <var> 2 MUL – multiply <var> with X register 3 DIV – divide <var> by X register 4 MOD – modulo divide <var> by X register 5 AND – logical and <var> with X register 6 OR – logical or <var> with X register 7 XOR – logical exor <var> with X register 8 NOT – copy logical inverted X register to <var> 9 LOAD – copy X register to <var> 10 SWAP – swap contents of <var> and X register 11 COMP – compare <var> with X register	0 <var> (0...255)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	the operand (don't care)

Example
Subtract X register from user variable #27.
Mnemonic: CALCVX SUB, 27

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of CALCVX SUB, 27	
Field	Value
Target address	01_h
Instruction number	$2B_h$
Type	01_h
Motor/Bank	$1B_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	48_h

Reply (Status=no error, value=0:	
Field	Value
Host address	02_h
Target address	01_h
Status	64_h
Instruction	$2B_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	92_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.40 CALCXV (Calculate using the X Register and a User Variable)

The CALCXV instruction modifies the X register using an arithmetical operation and the contents of a user variable. The parameters of this command are the arithmetic function and the index of a user variable (0...255). *This command is mainly intended for use in standalone mode.*

Related commands: CALCV, CALCAV, CALCVA, CALCVX, CALCVV.

Mnemonic: CALCXV <operation>, <var>

Binary representation

Binary Representation			
Instruction	Type	Motor/Bank	Value
44	0 ADD – add <var> to X register 1 SUB – subtract <var> from X register 2 MUL – multiply X register with <var> 3 DIV – divide X register by <var> 4 MOD – modulo divide X register by <var> 5 AND – logical and X register with <var> 6 OR – logical or X register with <var> 7 XOR – logical exor X register with <var> 8 NOT – copy logical inverted <var> to X register 9 LOAD – copy <var> to X register 10 SWAP – swap contents of <var> and X register 11 COMP – compare X register with <var>	0 <var> (0...255)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	the operand (don't care)

Example
Subtract user variable #27 from X register.
Mnemonic: CALCXV SUB, 27

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of CALCXV SUB, 27	
Field	Value
Target address	01_h
Instruction number	$2C_h$
Type	01_h
Motor/Bank	$1B_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	49_h

Reply (Status=no error, value=0:	
Field	Value
Host address	02_h
Target address	01_h
Status	64_h
Instruction	$2C_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	93_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.41 CALCV (Calculate using a User Variable and a Direct Value)

The CALCV directly modifies a user variable using an arithmetical operation and a direct value. This eliminates the need of using the accumulator register for such a purpose and thus can make the program shorter and faster. The parameters of this command are the arithmetic function, the index of a user variable (0…255) and a direct value. *This command is mainly intended for use in standalone mode.*

Related commands: CALCVA, CALCAV, CALCVX, CALCXV, CALCVV.

Mnemonic: CALCV <operation>, <var>, <value>

Binary representation

Binary Representation			
Instruction	Type	Motor/Bank	Value
45	0 ADD – add <value> to <var> 1 SUB – subtract <value> from <var> 2 MUL – multiply <var> with <value> 3 DIV – divide <var> by <value> 4 MOD – modulo divide <var> by <value> 5 AND – logical and <var> with <value> 6 OR – logical or <var> with <value> 7 XOR – logical exor <var> with <value> 8 NOT – logical invert <var> (<value> ignored) 9 LOAD – copy <value> to <var> 11 COMP – compare <var> with <value>	0 <var> (0…255)	<value>

Reply in Direct Mode	
Status	Value
100 - OK	the operand (don't care)

Example
Subtract 5000 from user variable #27.
Mnemonic: CALCV SUB, 27, 5000

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of CALCV SUB, 27, 5000	
Field	Value
Target address	01_h
Instruction number	$2D_h$
Type	01_h
Motor/Bank	$1B_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	13_h
Value (Byte 0)	88_h
Checksum	$E5_h$

Reply (Status=no error, value=5000:	
Field	Value
Host address	02_h
Target address	01_h
Status	64_h
Instruction	$2D_h$
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	13_h
Value (Byte 0)	88_h
Checksum	$2F_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.42 RST (Restart)

Stop the program, reset the TMCL interpreter and then re-start the program at the given label. This command can be used to re-start the TMCL program from anywhere in the program, also out of subroutines or interrupt rotuines. *This command is intended for standalone operation only.*

Internal function: The TMCL interpreter is reset (the subroutine stack, the interrupt stack and the registers are cleared) and then the program counter is set to the value passed to this command.

Related commands: JA, CSUB, STOP.

Mnemonic: RST <label>

Binary Representation			
Instruction	Type	Motor/Bank	Value
48	0 (don't care)	0 (don't care)	<restart address>

Example
Restart the program from a label, out of a subroutine:

```
Entry:
    MVP ABS, 0, 51200
    CSUB Subroutine
    ...
    ...
Subroutine:
    RST Entry
    RSUB
```

Binary form of the RST Entry command when the label Entry is at address 10:

Binary Form of RST Entry (assuming Entry at address 10)	
Field	Value
Target address	01_h
Instruction number	30_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	$0A_h$
Checksum	$3A_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.43 DJNZ (Decrement and Jump if not Zero)

Decrement a given user variable and jump to the given address if the user variable is greater than zero. This command can for example be used to easily program a counting loop, using any user variable as the loop counter. *This command is intended for standalone operation only.*

Internal function: The user variable passed to this command is decremented. If it is not zero then the TMCL program counter is set to the value passed to this command.

Related commands: JC, WAIT, CSUB.

Mnemonic: DJNZ <var>, <label>

Binary Representation			
Instruction	Type	Motor/Bank	Value
49	<user variable> (0…255)	0 (don't care)	<jump address>

Example
A counting loop in TMCL, using user variable #42:

```
    SGP 42, 2, 100
Loop:
    MVP ABS, REL, 51200
    WAIT POS, 0, 0
    WAIT TICKS, 0, 500
    DJNZ 42, Loop
```

Binary form of the DJNZ 42, Loop command when the label Loop is at address 1:

Binary Form of DJNZ Loop (assuming Loop at address 1)	
Field	Value
Target address	01_h
Instruction number	31_h
Type	64_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	01_h
Checksum	97_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.44 CALL (Conditional Subroutine Call)

The CALL command calls a subroutine in the TMCL program, but only if the specifed condition is met. Otherwise the program execution will be continued with the next command following the CALL command. The conditions refer to the result of a preceding comparison or assignment. *This command is intended for standalone operation only.*

Internal function: When the condition is met the actual TMCL program counter value will be saved to an internal stack. Afterwards the program counter will be overwritten with the address supplied to this command. 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, JC.

Mnemonic: CALL <condition>, <label>

Binary Representation			
Instruction	Type	Motor/Bank	Value
21	0 ZE - zero 1 NZ - not zero 2 EQ - equal 3 NE - not equal 4 GT - greater 5 GE - greater/equal 6 LT - lower 7 LE - lower/equal 8 ETO - time out error 9 EAL - external alarm 10 EDV - deviation error 11 EPO - position error	0 (don't care)	<jump address>

Example
Call a subroutine if a condition is met:

```
Loop:
    GIO 0, 1              //read analog value
    CALC SUB, 512         //subtract 512
    COMP 0                //compare with zero
    CALL LT, RunLeft      //Call routine "RunLeft" if accu<0
    CALL ZE, MotorStop    //Call routine "MotosStop" if accu=0
    CALL GT, RunRight     //Call routine "RunRight" if accu>0
    JA Loop

RunLeft:
    CALC MUL, -1
    ROLA 0
    RSUB
```

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

```
14
RunRight:
16    RORA 0
      RSUB
18
MotorStop:
20    GAP 2, 0
      JC ZE, MotorIsStopped
22    MST 0
MotorIsStopped:
24    RSUB
```

Binary form of CALL LT, RunLeft (assuming RunLeft at address 100)	
Field	Value
Target address	01_h
Instruction number	50_h
Type	06_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	64_h
Checksum	BB_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.45 MVPA (Move to Position specified by Accumulator Register)

With this command the motor will be instructed to move to a specified relative or absolute position. The contents of the accumulator register will be used as the target position. This command is non-blocking which means that 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 as well as other ramp parameters are defined by the appropriate axis parameters. For a list of these parameters please refer to section 4.
Positioning can be interrupted using MST, ROL or ROR commands.

Three operation types are available:

- Moving to an absolute position specified by the accumulator register contents.
- Starting a relative movement by means of an offset to the actual position.
- Moving the motor to a (previously stored) coordinate (refer to SCO for details).

Note The distance between the actual position and the new position must not be more than 2147483647 ($2^{31} - 1$) microsteps. Otherwise the motor will run in the opposite direction in order to take the shorter distance (caused by 32 bit overflow).

Internal function: Position mode is selected and the value stored in the accumulator register is copied to axis parameter #0 (target position).

Related commands: MVPXA, SAP, GAP, SCO, GCO, CCO, ACO, MST.

Mnemonic: MVPA <ABS|REL|COORD>, <axis>

Binary Representation			
Instruction	Type	Motor/Bank	Value
46	0 – ABS – absolute	0...5	0 (don't care)
	1 – REL – relative	0...5	0 (don't care)
	2 – COORD – coordinate	0...255	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Move motor 0 to position specified by accumulator.
Mnemonic: MVPA ABS, 0

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of MVPA ABS, 0	
Field	Value
Target address	01_h
Instruction number	$2E_h$
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$2F_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.46 ROLA (Rotate Left using the Accumulator Register)

Rotate in left direction (decreasing the position counter) using the velocity specified by the contents of the accumulator register. The velocity is given in internal units ([int]) of the TMC429 motion controller used on the TMCM-6110 module. Please see also section 6.1.

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

Related commands: RORA, MST, SAP, GAP.

Mnemonic: ROLA <axis>

Binary Representation			
Instruction	Type	Motor/Bank	Value
50	0 (don't care)	0...5	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Rotate left motor 0, velocity specified by accumulator.
Mnemonic: ROLA 0.

Binary Form of ROLA 0	
Field	Value
Target address	01_h
Instruction number	32_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	33_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.47 RORA (Rotate Right using the Accumulator Register)

Rotate in right direction (increasing the position counter) using the velocity specified by the contents of the accumulator register. The velocity is given in internal units ([int]) of the TMC429 motion controller used on the TMCM-6110 module. Please see also section 6.1.

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

Related commands: ROLA, MST, SAP, GAP.

Mnemonic: ROLA <axis>

Binary Representation			
Instruction	Type	Motor/Bank	Value
51	0 (don't care)	0...5	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Rotate right motor 0, velocity specified by accumulator.
Mnemonic: RORA 0.

Binary Form of RORA 0	
Field	Value
Target address	01_h
Instruction number	33_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	33_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.48 MVPXA (Move to Position specified by Accumulator Register and X Register)

With this command the motor will be instructed to move to a specified relative or absolute position. The contents of the accumulator register will be used as the target position, and the contents of the X register specifies the motor number. This command is non-blocking which means that 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 as well as other ramp parameters are defined by the appropriate axis parameters. For a list of these parameters please refer to section 4.
Positioning can be interrupted using MST, ROL or ROR commands.

Three operation types are available:

- Moving to an absolute position specified by the accumulator register contents.
- Starting a relative movement by means of an offset to the actual position.
- Moving the motor to a (previously stored) coordinate (refer to SCO for details).

Note The distance between the actual position and the new position must not be more than 2147483647 ($2^{31} - 1$) microsteps. Otherwise the motor will run in the opposite direction in order to take the shorter distance (caused by 32 bit overflow).

Internal function: The value stored in the accumulator is copied to the target position of the axis specified by the X register.

Related commands: MVPA, SAP, GAP, SCO, GCO, CCO, ACO, MST.

Mnemonic: MVPXA <ABS|REL|COORD>

Binary Representation			
Instruction	Type	Motor/Bank	Value
47	0 – ABS – absolute	0 (don't care)	0 (don't care)
	1 – REL – relative	0 (don't care)	0 (don't care)
	2 – COORD – coordinate	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Move motor specified by X register to position specified by accumulator.
Mnemonic: MVPXA ABS

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of MVPXA ABS	
Field	Value
Target address	01_h
Instruction number	$2F_h$
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	30_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.49 ROLXA (Rotate Left using the Accumulator Register and X Register)

Rotate in left direction (decreasing the position counter) using the velocity specified by the contents of the accumulator register and the motor specified by the contents of the X register. The velocity is given in internal units ([int]) of the TMC429 motion controller used on the TMCM-6110 module. Please see also section 6.1.

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

Related commands: RORXA, MSTX, SAP, GAP.

Mnemonic: ROLXA

Binary Representation			
Instruction	Type	Motor/Bank	Value
52	0 (don't care)	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Rotate left motor specified by X register, velocity specified by accumulator.
Mnemonic: ROLXA.

Binary Form of ROLXA	
Field	Value
Target address	01_h
Instruction number	34_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	35_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.50 RORXA (Rotate Right using the Accumulator Register and X Register)

Rotate in right direction (increasing the position counter) using the velocity specified by the contents of the accumulator register and the motor specified by the contents of the X register. The velocity is given in internal units ([int]) of the TMC429 motion controller used on the TMCM-6110 module. Please see also section 6.1.

Internal function: Velocity mode is selected. Then, the velocity value is transferred to the target velocity (axis parameter #2).
Related commands: ROLXA, MSTX, SAP, GAP.

Mnemonic: RORXA

Binary Representation			
Instruction	Type	Motor/Bank	Value
53	0 (don't care)	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Rotate left motor specified by X register, velocity specified by accumulator.
Mnemonic: RORXA.

Binary Form of RORXA	
Field	Value
Target address	01_h
Instruction number	35_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	36_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.51 MSTX (Motor Stop using X Register)

The motor specified by the X register is instructed to stop using a soft stop.

Internal function: For the axis specified by the X register, velocity mode is selected and the target speed is set to zero.

Related commands: RORXA, ROLXA.

Mnemonic: MSTX

Binary Representation			
Instruction	Type	Motor/Bank	Value
54	0 (don't care)	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Stop motor specified by X register.
Mnemonic: MSTX.

Binary Form of MSTX	
Field	Value
Target address	01_h
Instruction number	36_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	37_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.52 SAPX (Set Axis Parameter, Axis specified by X Register)

The SAPX command nearly works the same way as the SAP command, except that the axis number is specified by the X register. *This command is mainly intended for use in standalone mode.*

Info	For a table with parameters and values which can be used together with this command please refer to section 4.

Internal function: The specified value is written to the axis parameter specified by the parameter number.
Related commands: GAPX, AAPX.

Mnemonic: SAPX <parameter number>, <value>

Binary representation

Binary Representation			
Instruction	Type	Motor/Bank	Value
16	see chapter 4	(don't care)	<value>

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example Set the maximum positioning speed for the motor specified by the X register to 1000.
Mnemonic: SAPX 4, 1000.

Binary Form of SAPX 4, 1000	
Field	Value
Target address	01_h
Instruction number	10_h
Type	04_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	03_h
Value (Byte 0)	$E8_h$
Checksum	00_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.53 GAPX (Get Axis Parameter, Axis specified by X Register)

The GAPX command nearly works the same way as the GAP command, except that the motor number is specified by the X register. *This command is mainly intended for use in standalone mode.*

Info	For a table with parameters and values that can be used together with this command please refer to section 4.

Internal function: The specified value gets copied to the accumulator.
Related commands: SAPX, AAPX.

Mnemonic: GAPX <parameter number>

Binary Representation			
Instruction	Type	Motor/Bank	Value
17	see chapter 4	(don't care)	<value>

Reply in Direct Mode	
Status	Value
100 - OK	value read by this command

Example
Get the actual position of the motor specified by the X register.
Mnemonic: GAPX 1.

Binary Form of GAPX 1	
Field	Value
Target address	01_h
Instruction number	11_h
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	13_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.54 AAPX (Accu to Axis Parameter, Axis specified by X Register)

The AAPX command nearly works the same way as the AAP command, with the only exception that the axis number is specified by the X register. *This command is mainly intended for use in standalone mode.*

Info	For a table with parameters and values which can be used together with this command please refer to section 4.

Related commands: SAPX, GAPX.

Mnemonic: AAPX <parameter number>

Binary Representation			
Instruction	Type	Motor/Bank	Value
18	see chapter 4	(don't care)	(don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Loop over all motors, positon them at different speeds:

```
    SGP 0, 2, 5      //initalize user variable 0 with 5 (used as loop counter)
    CALCXV LOAD, 0   //set X register to 0
    SGP 1, 2, 50     //initialize user variable 1 with 50

Loop:
    GGP 1, 2             //copy user variable 1 to accu
    CALCXV LOAD, 0       //copy user variable 0 (loop counter) to X register
    AAPX 4               //set maximum positioning speed of motor
    CALC LOAD, 51200     //initialize accu with target position
    MVPXA ABS            //position motor
    WAIT TICKS, 0, 100   //wait 1 second
    CALCV MUL, 1, 2      //multiply user variable 1 with 2
    DJNZ 0, Loop         //next iteration
```

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Binary Form of AAP 0	
Field	Value
Target address	01_h
Instruction number	12_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	13_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.55 SIV (Set Indexed Variable)

This command copies a direct value to a TMCL user variable. The index of the user variable (0…255) is specified by the content of the X register. Therefore the value in the X register must not be lower than zero or greater than 255. Otherwise this command will be ignored. *This command is mainly intended for use in standalone mode.*

Internal function: The direct value supplied to this command will be copied to the user variable specified by the X register.

Related commands: AIV, GIV.

Mnemonic: SIV

Binary Representation			
Instruction	Type	Motor/Bank	Value
55	0 (don't care)	0 (don't care)	<value>

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Copy the value 3 to the user variable indexed by the X register.
Mnemonic: SIV 3.

Binary Form of SIV 3	
Field	Value
Target address	01_h
Instruction number	37_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	03_h
Checksum	$3B_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.56 GIV (Get Indexed Variable)

This command reads a TMCL user variable and copies its content to the accumulator register. The index of the user variable (0...255) is specified by the X register. Therefore the content of the X register must not be lower than zero or greater than 255. Otherwise this command will be ignored. *This command is mainly intended for use in standalone mode.*

Internal function: The user variable specified by the x register will be copied to the accumulator register.

Related commands: SIV, AIV.

Mnemonic: GIV

Binary Representation			
Instruction	Type	Motor/Bank	Value
55	0 (don't care)	0 (don't care)	0 (don't care)

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Read the user variable indexed by the X register.
Mnemonic: GIV.

Binary Form of GIV	
Field	Value
Target address	01_h
Instruction number	38_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	03_h
Checksum	39_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.57 AIV (Accumulator to Indexed Variable)

This command copies the content of the accumulator to a TMCL user variable. The index of the user variable (0...255) is specified by the content of the X register. Therefore the value in the X register must not be lower than zero or greater than 255. Otherwise this command will be ignored. *This command is mainly intended for use in standalone mode.*

Internal function: The accumulator will be copied to the user variable specified by the X register.

Related commands: SIV, GIV.

Mnemonic: AIV

Binary Representation			
Instruction	Type	Motor/Bank	Value
55	0 (don't care)	0 (don't care)	<value>

Reply in Direct Mode	
Status	Value
100 - OK	don't care

Example
Copy the accumulator to the user variable indexed by the X register.
Mnemonic: AIV.

Binary Form of AIV	
Field	Value
Target address	01_h
Instruction number	39_h
Type	00_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	00_h
Checksum	$3A_h$

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.58 Customer specific Command Extensions (UF0...UF7 – User Functions)

These commands are used for customer specific extensions of TMCL. They will be implemented in C by Trinamic. Please contact the sales department of Trinamic Motion Control GmbH & Co KG if you need a customized TMCL firmware.

Related commands: none.

Mnemonic: UF0...UF7

Binary Representation			
Instruction	Type	Motor/Bank	Value
64...71	<user defined>	0 <user defined>	0 <user defined>

Reply in Direct Mode	
Status	Value
100 - OK	user defined

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.59 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 a motor has reached its target position.

Related commands: none.

Binary Representation			
Instruction	Type	Motor/Bank	Value
138	0/1	0 (don't care)	<bit vector>

With command 138 the value field is a bit vector. It shows for which motors one would like to have a position reached message. The value field contains a bit mask where every bit stands for one motor. Bit 0 stands for motor #0, bit 1 stands for motor #1, bit 2 stands for motor #2 and so on. With the type field set to 0, only for the next MVP command that follows this command a position reached message will be generated. With type set to 1 a position reached message will be generated for every MVP command that follows this command. It is recommended to use the latter option.

Example
Get target position reached message for motors #2 and #3.

Binary Form for this example	
Field	Value
Target address	01_h
Instruction number	$8A_h$
Type	01_h
Motor/Bank	00_h
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	05_h
Checksum	91_h

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Reply in Direct Mode	
Field	Value
Target address	01_h
Host address	02_h
Status	64_h (100)
Command	$8A_h$ (138)
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	Motor bit mask
Checksum	depends also on motor bit mask

Additional Reply after Motor has reached Target Position	
Field	Value
Target address	01_h
Host address	02_h
Status	80_h (128)
Command	$8A_h$ (138)
Value (Byte 3)	00_h
Value (Byte 2)	00_h
Value (Byte 1)	00_h
Value (Byte 0)	Motor bit mask
Checksum	depends also on motor bit mask

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

3.7.60 TMCL Control Commands

There is a set of TMCL commands which are called TMCL control commands. These commands can only be used in direct mode and not in a standalone program. For this reason they only have opcodes, but no mnemonics. Most of these commands are only used by the TMCL-IDE (in order to implement e.g. the debugging functions in the TMCL creator). Some of them are also interesting for use in custom host applications, for example to start a TMCL routine on a module, when combining direct mode and standalone mode (please see also section 8.6. The following table lists all TMCL control commands.

The motor/bank parameter is not used by any of these functions and thus is not listed in the table. It should always be set to 0 with these commands.

TMCL Control Commands			
Instruction	Description	Type	Value
128 – stop application	stop a running TMCL application	0 (don't care)	0 (don't care)
129 – run application	start or continue TMCL program execution	0 – from current address 1 – from specific address	0 (don't care) starting address
130 – step application	execute only the next TMCL command	0 (don't care)	0 (don't care)
131 – reset application	Stop a running TMCL program. Reset program counter and stack pointer to zero. Reset accumulator and X register to zero. Reset all flags.	0 (don't care)	0 (don't care)
132 – enter download mode	All following commands (except control commands) are not executed but stored in the TMCL memory.	0 (don't care)	start address for download
133 – exit download mode	End the download mode. All following commands are executed normally again.	0 (don't care)	0 (don't care)
134 – read program memory	Return contents of the specified program memory location (special reply format).	0 (don't care)	address of memory location

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Instruction	Description	Type	Value
135 – get application status	Return information about the current status, depending on the type field.	0 - return mode, wait flag, memory pointer 1 - return mode, wait flag, program counter 2 - return accumulator 3 - return X register	0 (don't care)
136 – get firmware version	Return firmware version in string format (special reply) or binary format).	0 - string format 1 - binary format	0 (don't care)
137 – restore factory settings	Reset all settings in the EEPROM to their factory defaults. This command does not send a reply.	0 (don't care)	set to 1234
255 – software reset	Restart the CPU of the module (like a power cycle). The reply of this command might not always get through.	0 (don't care)	set to 1234

Table 14: TMCL Control Commands

Especially the commands 128, 129, 131, 136 and 255 are interesting for use in custom host applications. The other control commands are mainly being used by the TMCL-IDE.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

4 Axis Parameters

Most motor controller features of the TMCM-6110 module are controlled by axis parameters. Axis parameters can be modified or read using SAP, GAP and AAP commands. Some axis parameters can also be stored to or restored from the EEPROM using STAP and RSAP commands. This chapter describes all axis parameters that can be used on the TMCM-6110 module. Please see also section 6.1 for information about converting between internal units of the module (named [int] in this manual) and real world units.

There are different parameter access types, like read only or read/write. Table 15 shows the different parameter access types used in the axis parameter tables.

Meaning of the Letters in the Access Column		
Access type	Command	Description
R	GAP	Parameter readable
W	SAP, AAP	Parameter writable
E	STAP, RSAP	Parameter can be stored in the EEPROM

Table 15: Meaning of the Letters in the Access Column

All Axis Parameters of the TMCM-6110 Module				
Number	Axis Parameter	Description	Range [Units]	Access
0	Target position	The desired target position in position mode	-2147483648 …2147483647 [μsteps]	RW
1	Actual position	The actual position of the motor. Stop the motor before overwriting it. Should normally only be overwritten for reference position setting.	-2147483648 …2147483647 [μsteps]	RW
2	Target speed	The desired speed in velocity mode. Not valid in position mode.	-2047 …2047 [int]	RW
3	Actual speed	The actual speed of the motor.	-2047 …2047 [int]	R
4	Maximum positioning speed	The maximum speed used for positioning ramps.	1…2047 [int]	RWE
5	Maximum acceleration	Maximum acceleration during ramp-up and maximum deceleration during ramp-down.	1…2047 [int]	RWE

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Axis Parameter	Description	Range [Units]	Access
6	Maximum current	Motor current used when motor is running. The maximum value is 255 which means 100% of the maximum current of the module. The current 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 setting, as too high values can cause motor damage.*	0…255	RW
7	Standby current	The current used when the motor is not running. The maximum value is 255 which means 100% of the maximum current of the module. This value should be as low as possible so that the motor can cool down when it is not moving. Please see also parameter 214.	0…255	RW
8	Position reached flag	This flag is always set when target position and actual position are equal.	0/1	R
9	Home switch state	The logical state of the home switch input.	0/1	R
10	Right limit switch state	The logical state of the right limit switch input.	0/1	R
11	Left limit switch state	The logical state of the left limit switch input.	0/1	R
12	Right limit switch disable	0 - switch activated 1 - switch deactivated	0/1	RWE
13	Left limit switch disable	0 - switch activated 1 - switch deactivated	0/1	RWE
130	Minimum speed	This is the stop speed (the start speed is always 1). Default value is 1. Can be set higher in order to reach the target position faster. Never set lower than 1, as in this case the target position cannot be reached.	1…2047 [int]	RWE
135	Actual acceleration	The current acceleration.	-2047…2047 [int]	R

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Axis Parameter	Description	Range [Units]	Access
138	Ramp mode	Automatically set when using ROR, ROL, MST and MVP. 0: Position mode. Steps are generated, when the parameters actual position and target position differ. Trapezoidal speed ramps are provided. 2: Velocity mode. The motor will run continuously and the speed will be changed using the maximum acceleration parameter when the target speed gets changed. For special purposes, the soft mode (value 1) with exponential decrease of speed can be selected.	0/1/2	RW
140	Microstep resolution	Microsteps per full step: 0 fullstep 1 halfstep 2 4 microsteps 3 8 microsteps 4 16 microsteps 5 32 microsteps 6 64 microsteps 7 128 microsteps 8 256 microsteps **It is strongly recommended to use at least 8 microsteps.**	0…8	RW
149	Soft stop flag	0 - Hard stop when a stop switch is hit. 1 - Soft stop (using deceleration ramp) when a stop switch is hit.	0/1	RWE
150	End switch power down mode	0 - Stay at run current when stopped by end switch. 1 - Use stand by current when stopped by end switch.	0/1	RW
153	Ramp divisor	The exponent of the scaling factor for the ramp generator. Change this parameter carefully (in steps of one) and only while the motor is not moving. Lower values lead to higher accelerations. This parameter specifies the relation between internal and real world acceleration units.	0…13	RWE

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Axis Parameter	Description	Range [Units]	Access
154	Pulse divisor	The exponent of the scaling factor for the pulse (step) generator. Change this parameter carefully (in steps of one) and only while the motor is not moving. Lower values lead to higher speeds. This parameter specifies the relation between internal and real world velocity units.	0…13	RWE
160	Step interpolation enable	Step interpolation is supported with 16 microstep setting only. With this option activated, each microstepstep will internally be executed as 16 1/256 microsteps. This causes the motor to run as smooth as with 256 microsteps resolution. 0 - step interpolation off 1 - step interpolation on	0/1	RW
161	Double step enable	With this option turned on, each microstep will be executed twice. Normally, use this option only together with the step/direction input. Every edge of the step signal then causes a microstep to be executed. 0 - double step off 1 - double step on	0/1	RW
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. Normally leave at the default value.	0…3	RW
163	Constant TOff mode	Selection of the chopper mode: 0 – spread cycle 1 – classic constant off time	0/1	RW
164	Disable fast decay comperator	See parameter 163. For ”classic const. off time” setting this parameter to ”1” will disable current comparator usage for termination of fast decay cycle.	0/1	RW
165	Chopper hysteresis end / fast decay time	See parameter 163. For ”spread cycle” chopper mode this parameter will set / return the hysteresis end setting (hysteresis end value after a number of decrements). For ”classic const. off time” chopper mode this parameter will set / return the fast decay time.	0…15	RW
166	Chopper hysteresis start / sine wave offset	See parameter 163. For ”spread cycle” chopper mode this parameter will set / return the Hysteresis start setting (please note that this value is an offset to the hysteresis end value). For ”classic const. off time” chopper mode this parameter will set / return the sine wave offset.	0…8	RW

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Axis Parameter	Description	Range [Units]	Access
167	Chopper off time (TOff)	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: $N_{CLK} = 12 + 32 * tOFF$ (Minimum is 64 clocks) Setting this parameter to zero completely disables all driver transistors and the motor can free-wheel.	0…15	RW
168	SmartEnergy current minimum (SEIMIN)	Sets the lower motor current limit for CoolStep operation by scaling the maximum current (see axis parameter 6) value. Minimum motor current: 0 - $\frac{1}{2}$ of CS 1 - $\frac{1}{4}$ 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: $([AP172] + 1) * 32$ Upper StallGuard threshold: $([AP172] + [AP170] + 1) * 32$	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	0…3	RW
172	SmartEnergy hysteresis start	The lower threshold for the StallGuard2 value (see SmartEnergy current up step). **Setting this to 0 (default) turns off CoolStep.**	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

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Axis Parameter	Description	Range [Units]	Access
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.	-64...+63	RW
175	Slope control high side	Determines the slope of the motor driver outputs. Leave at default value unless differently recommended by Trinamic customer support. 0 - lowest slope 3 - fastest slope	0...3	RW
176	Slope control low side	Determines the slope of the motor driver outputs. Leave at default value unless differently recommended by Trinamic customer support. 0 - lowest slope 3 - fastest slope	0...3	RW
177	Short protection disable	Switches short to ground protection of the motor driver on or off. Leave at default value unless differently recommended by Trinamic customer support. 0 - Short to GND protection on 1 - Short to GND protection off	0/1	RW
178	Short detection timer	Timer value for short circuit protection of the motor driver. Leave at default value unless differently recommended by Trinamic customer support. 0 - 3.2μs 1 - 1.6μs 2 - 1.2μs 3 - 0.8μs	0...3	RW
179	Vsense	Sense resistor voltage based current scaling. 0 - Full scale sense resistor voltage is 1/18 VDD 1 - Full scale sense resistor voltage is 1/36 VDD Leave at default value. Do not change!	0/1	R
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	R
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...2047 [int]	RW
182	SmartEnergy threshold speed	Above this speed CoolStep becomes enabled.	0...2047 [int]	RW

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Axis Parameter	Description	Range [Units]	Access
183	SmartEnergy slow run current	Sets the motor current which is used below the threshold speed. A value of 255 means 100% of the maximum current of the module.	0…255	RW
184	Random TOff mode	0 - Chopper off time is fixed 1 - Chopper off time is random	0/1	RW
193	Reference 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. Additional functions: • Add 128 to a mode value for inverting the home switch (can be used with mode 5…8). • Add 64 to a mode for searching the right instead of the left reference switch (can be used with mode 1…4).	1…8	RW
194	Reference search speed	This value specifies the speed for roughly searching the reference switch.	0…2047 [int]	RW
195	Reference switch speed	This parameter specifies the speed for searching the switching point. It should be slower than parameter 194.	0…2047 [int]	RW
196	End switch distance	This parameter provides the distance between the end switches after executing the RFS command (with reference search mode 2 or 3).	-2147483648 …2147483647 [μsteps]	R
197	Last reference position	This parameter contains the last position value before the position counter is set to zero during reference search.	-2147483648 …2147483647 [μsteps]	R

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Axis Parameter	Description	Range [Units]	Access
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. Same scaling as with axis parameter #6.	0…255	RW
204	Freewheeling	Time after which the power to the motor will be cut when its velocity has reached zero (a value of 0 (default setting) means never).	0…65535 [10ms]	RWE
206	Actual load value	Readout of the actual load value used for stall detection (StallGuard2).	0…1023	R
207	Extended error flags	0 - no error 1 - StallGuard error This error flag will be cleared automatically when reading this parameter or when executing a motion command.	0…1	R
208	TMC262 error flags	A combination of the following values: Bit 0: StallGuard2 status (1: threshold reached) Bit 1: Overtemperature (1: driver is shut down due to overtemperature) Bit 2: Overtemperature pre-warning (1: temperature threshold is exceeded) Bit 3: Short to ground A (1: short condition detected, 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) Please also refer to the TMC262 Datasheet for more information.	0…255	R
213	Group index	All motors on the module which have the same group index will get the same commands when a ROL, ROR, MST, MVP or RFS is issued for one of these motors. Setting the group index to 0 (default) turns off this feature for the motor.	0…255	RW
214	Power down delay	Standstill period before the motor current will be switched to standby current. The default value is 200 which means 2000ms.	1…65535 [10ms]	RWE

Table 16: All Axis Parameters of the TMCM-6110 Module

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

5 Global Parameters

The following sections describe all global parameters that can be used with the SGP, GGP, AGP, STGP and RSGP commands. Global parameters are grouped into banks:

- Bank 0: Global configuration of the module.
- Bank 1: Not used.
- Bank 2: TMCL user variables.
- Bank 3: TMCL interrupt configuration.

5.1 Bank 0

Parameters with numbers from 64 on configure all settings that affect the overall behaviour of a module. These are things like the serial address, the RS485 baud rate or the CAN bit rate (where appropriate). 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 automatically stored in the EEPROM.

Note

- 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.
- Some configurations of the interface (for example baud rates that are not supported by the PC) may leed to the fact that the module cannot be reached any more. In such a case please see the TMCM-6110 Hardware Manual on how to reset all parameters to factory default settings.
- Some settings (especially interface bit rate settings) do not take effect immediately. For those settings, power cycle the module after changing them to make the changes take effect.

There are different parameter access types, like read only or read/write. Table 17 shows the different parameter access types used in the global parameter tables.

Meaning of the Letters in the Access Column		
Access type	Command	Description
R	GGP	Parameter readable
W	SGP, AGP	Parameter writable
E	STGP, RSGP	Parameter can be stored in the EEPROM
A	SGP	Automatically stored in the EEPROM

Table 17: Meaning of the Letters in the Access Column

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

<table>
<tr><th colspan="5">All Global Parameters of the TMCM-6110 Module in Bank 0</th></tr>
<tr><th>Number</th><th>Global Parameter</th><th colspan="3">Description</th><th>Range [Units]</th><th>Access</th></tr>
<tr><td>65</td><td>RS485 baud rate</td><td colspan="3"><table><tr><td>0</td><td>9600</td><td>Default</td></tr><tr><td>1</td><td>14400</td><td></td></tr><tr><td>2</td><td>19200</td><td></td></tr><tr><td>3</td><td>28800</td><td></td></tr><tr><td>4</td><td>38400</td><td></td></tr><tr><td>5</td><td>57600</td><td></td></tr><tr><td>6</td><td>76800</td><td></td></tr><tr><td>7</td><td>115200</td><td></td></tr><tr><td>8</td><td>230400</td><td></td></tr></table></td><td>0…8</td><td>RWA</td></tr>
<tr><td>66</td><td>Serial address</td><td colspan="3">Module (target) address for RS485.</td><td>1…255</td><td>RWA</td></tr>
<tr><td>67</td><td>ASCII Mode</td><td colspan="3">Configure the TMCL ASCII interface:<table><tr><td>Bit 0</td><td>0 - start up in binary (normal) mode
1 - start up in ASCII mode</td></tr><tr><td>Bits 4 & 5</td><td>00 - Echo back each character
01 - Echo back complete command
10 - Do not send echo, only send command reply</td></tr></table></td><td>0…63</td><td>RWA</td></tr>
<tr><td>68</td><td>Serial heartbeat</td><td colspan="3">Serial heartbeat for RS485 interface and USB interface. If this time limit is up and no further command is received by the module the motor will be stopped. Setting this parameter to 0 (default) turns off the serial heartbeat function.</td><td>0…65535</td><td>RWA</td></tr>
<tr><td>69</td><td>CAN bit rate</td><td colspan="3"><table><tr><td>2</td><td>20kBit/s</td></tr><tr><td>3</td><td>50kBit/s</td></tr><tr><td>4</td><td>100kBit/s</td></tr><tr><td>5</td><td>125kBit/s</td></tr><tr><td>6</td><td>250kBit/s</td></tr><tr><td>7</td><td>500kBit/s</td></tr><tr><td>8</td><td>1000kBit/s (Default)</td></tr></table></td><td>2…8</td><td>RWA</td></tr>
<tr><td>70</td><td>CAN reply ID</td><td colspan="3">The CAN ID for replies from the board (default: 2).</td><td>0…2047</td><td>RWA</td></tr>
<tr><td>71</td><td>CAN ID</td><td colspan="3">The module (target) address for CAN (default: 1).</td><td>0…2047</td><td>RWA</td></tr>
</table>

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Global Parameter	Description	Range [Units]	Access
75	Telegram pause time	Pause time before the reply via RS485 is sent. For use with older RS485 interfaces it is often necessary to set this parameter to 15 or more (e.g. RS485 adapters controlled by the RTS pin). For CAN interface this parameter has no effect!	0...255 [ms]	RWA
76	Serial host address	Host address used in the reply telegrams sent back via RS485.	0...255	RWA
77	Auto start mode	0 - Do not start TMCL application after power up (default). 1 - Start TMCL application automatically after power up.	0/1	RWA
79	End switch polarity	0: normal polarity for motor 0...5 1: reverse polarity for motor 0...2, normal polarity for motor 3...5 2: normal polarity for motor 0...2, reverse polarity for motor 3...5 3: reverse polarity for motor 0...5	0...3	RWA
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 **When switching off the protection against disassembling (changing this parameter from 1 or 3 to 0 or 2, the program will be erased first!**	0/1/2/3	RWA
82	CAN heartbeat	Heartbeat for CAN interface. If this time limit is up and no further command is received the motor will be stopped. Setting this parameter to 0 (default) turns off the CAN heartbeat function.	0...65535	RWA
83	CAN secondary address	Second CAN ID for the module. Switched off when set to zero. See section 7.5.	0...2047	RWA
84	Coordinate storage	0 - coordinates are stored in RAM only (but can be copied explicitly between RAM and EEPROM) 1 - coordinates are always also stored in the EEPROM and restored on startup	0/1	RWA
85	Do not restore user variables	Determines if TMCL user variables are to be restored from the EEPROM automatically on startup. 0 - user variables are restored (default) 1 - user variables are not restored	0/1	RWA

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Global Parameter	Description	Range [Units]	Access
87	Serial secondary address	Second module (target) address for RS485. Setting this parameter to 0 switches off the seconndary address. See section 7.4.	0...255	RWA
90	Reverse shaft	Reverse motor direction. 0: normal direction for motor 0...5 1: reverse direction for motor 0...2, normal direction for motor 3...5 2: normal direction for motor 0...2, reverse direction for motor 3...5 3: reverse direction for motor 0...5	0...3	RWA
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	Contains the address of the currently executed TMCL command.		R
132	TMCL tick timer	A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value.	0...2147483647	RW
133	Random number	Returns a random number. The seed value can be set by writing to this parameter.	0...2147483647	RW
255	Suppress reply	The reply in direct mode will be suppressed when this parameter is set to 1. This parameter cannot be stored to EEPROM and will be reset to 0 on startup. The reply will not be suppressed for GAP, GGP and GIO commands.	0/1	RW

Table 18: All Global Parameters of the TMCM-6110 Module in Bank 0

5.2 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 these variables form the interface between extensions of the firmware (written by Trinamic in C) and TMCL applications.

5.3 Bank 2

Bank 2 contains general purpose 32 bit variables for use in TMCL applications. They are located in RAM and the first 56 variables can also be stored permanently in the EEPROM. After booting, their values are automatically restored to the RAM. Up to 256 user variables are available. Please see table 17 for an explanation of the different parameter access types.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

User Variables in Bank 2				
Number	Global Parameter	Description	Range [Units]	Access
0…55	user variables #0…#55	TMCL user variables	-2147483648 … 2147483647	RWE
56…255	user variables #56…#255	TMCL user variables	-2147483648 … 2147483647	RWE

Table 19: User Variables in Bank 2

5.4 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.**
Table 20 shows all interrupt parameters that can be set. Please see table 17 for an explanation of the parameter access types.

Interrupt Parameters in Bank 3				
Number	Global Parameter	Description	Range [Units]	Access
0	Timer 0 period (ms)	Time between two interrupts	0…4294967295 [ms]	RW
1	Timer 1 period (ms)	Time between two interrupts	0…4294967295 [ms]	RW
2	Timer 2 period (ms)	Time between two interrupts	0…4294967295 [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
29	Stop left 1 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0…3	RW
30	Stop right 1 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0…3	RW
31	Stop left 2 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0…3	RW
32	Stop right 2 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0…3	RW
33	Stop left 3 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0…3	RW
34	Stop right 3 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0…3	RW
35	Stop left 4 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0…3	RW

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Global Parameter	Description	Range [Units]	Access
36	Stop right 4 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
37	Stop left 5 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
38	Stop right 5 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
39	Input 0 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
40	Input 1 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
41	Input 2 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
42	Input 3 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
43	Input 4 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
44	Input 5 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
45	Input 6 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW
46	Input 7 trigger transition	0=off, 1=low-high, 2=high-low, 3=both	0...3	RW

Table 20: Interrupt Parameters in Bank 3

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

6 Module Specific Hints

This section contains some hints that are specific to the TMCM-6110 module.

6.1 Velocity and Acceleration Calculation

On the TMCM-6110 module, the internal units of the TMC429 motion controller IC are used to specify velocity and acceleration values. The formulae for converting between internal units of this module and real world units are given in this section.

Note The Parameter Calculator tool in the TMCL-IDE can be used to easily convert between all velocity and acceleration units. So it is normally not necessary to do the following calculations manually.

6.1.1 Velocity Conversion

Axis parameters #2, #3, #4, #130, #181, #182 , #194 and #195 use internal units ([int], value range 0...2047 resp. -2047...2047). Axis parameter #154 (pulse divisor) defines the relation between internal velocity units and real world velocity units (pulses (microsteps) per second [pps]).
The formula for conversion from internal velocity units to pps is:

$$v_{pps} = \frac{16 \cdot 10^6 \cdot v_{int}}{2^{pulse_div} \cdot 2048 \cdot 32} \tag{1}$$

where v_{int} is the velocity value in internal units (-2047...2047) and $pulse_div$ is the pulse divisor (0...13, axis parameter #154). The result v_{pps} is the resulting microstep frequency.
For example, a velocity value of 1678 ($v_{int} = 1678$) and a pulse divisor of 3 ($pulse_div = 3$) will result in a microstep frequency of approximately 51208.5pps.
To convert from microstep frequency to fullstep frequency, the microstep resolution (parameter #140) has to be taken into account. Just divide the microstep frequency by the microstep resolution to convert this into fullstep frequency:

$$f_{full} = \frac{f_\mu}{2^{ap_{140}}} \tag{2}$$

where f_{full} is the fullstep frequency, f_μ is the microstep frequency (same as v_{pps} in equation (1)), and ap_{140} is the value of axis parameter #140 (0...8).

6.1.2 Acceleration Conversion

Axis parameter #5 uses internal units ([int], value range 0...2047) to define the acceleration ramp. Axis parameters #153 (ramp divisor) and #154 (pulse divisor) define the relation between internal acceleration units and real world acceleration units (pps per second [pps^2]).
The formula for conversion from internal acceleration units to pps^2 is:

$$a_{pps} = \frac{(16 \cdot 10^6)^2 \cdot a_{int}}{2^{ramp_div+pulse_div+29}} \tag{3}$$

where a_{pps} is the acceleration in pps^2 and a_{int} is the acceleration value in internal units. The value $ramp_div$ is axis parameter #153 and the value $pulse_div$ is axis parameter #154.
For example, an acceleration value of 100 ($a_{int} = 100$) together with a ramp divisor of 7 ($ramp_div = 7$) and a pulse divisor of 3 ($pulse_div = 3$) will result in an acceleration of approximately 46566pps^2.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

6.1.3 How to choose Pulse Divisor and Ramp Divisor

The pulse divisor should be chosen in such a way that the range of the velocity internal units (0...2047) covers the entire velocity range which is needed by the application. Ideally, set the pulse divisor so that the maximum velocity value of 2047 leads to a velocity that is just little but not much above the maximum velocity needed by your application. This way it is possible to make optimum use of the internal velocity resolution. The same is true for the acceleration and the ramp divisor.
The difference between ramp divisor and pulse divisor may limit the usable acceleration range for positioning ramps (MVP command), but not for velocity mode (ROL/ROR command). These limits are given by the following equations:

$$\begin{aligned} a_{lower_limit} &= 2^{ramp_div-pulse_div-1} \\ a_{upper_limit} &= 2^{ramp_div-pulse_div+12} - 1 \end{aligned}$$

If a_{upper_limit} should be greater than 2047 then the upper limit is 2047.

6.1.4 Conversion between PPS, RPM and RPS

In order to convert between pps units and units like rounds per second (rps) or rounds per minute (rpm), one has to know the fullstep resolution of the motor (full steps per round) and the microstep resolution setting of the module (axis parameter #140, default setting is 256 microsteps per full step).
So to convert from pps to rps, use the following formula:

$$v_{rps} = \frac{v_{pps}}{r_{fullstep} \cdot r_{microstep}}$$

To convert from rps to rpm, use:

$$v_{rpm} = v_{rps} \cdot 60$$

With the following symbols:

- v_{rps}: velocity in rounds per second
- v_{rpm}: velocity in rounds per minute
- v_{pps}: velocity in pulses (microsteps) per second
- $r_{fullstep}$: fullstep resolution of the motor (with most motors 200 (1.8°))
- $r_{microstep}$: microstep setting of the module (default 256)

So, with a 200 fullsteps motor and a microstep setting of 256 (axis parameter #140 = 8), a velocity of 51200pps will result in 1rps (60rpm).

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

7 Hints and Tips

This chapter gives some hints and tips on using the functionality of TMCL, for example how to use and parameterize the built-in reference search algorithm. You will also find basic information about StallGuard2™ and CoolStep™ in this chapter.

7.1 Reference Search

The built-in reference search features switching point calibration and support for a home switch and/or one or two end switches. The internal operation is based on a state machine that can be started, stopped and monitored (instruction RFS, opcode 13). The settings of the automatic stop functions corresponding to the end switches (axis parameters 12 and 13) do not influence the reference search.

Notes:

- Until the reference switch is found for the first time, the searching speed set by axis parameter 194 is used.
- After hitting the reference 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. The speed used for this calibration is defined by axis parameter 195.

Axis parameter 193 defines the reference search mode to be used. Choose one of the reference search modes shown in table 21 and in the following subsections:

Reference Search Modes	
Value	Description
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 negative direction, ignore end switches
8	search home switch in positive direction, ignore end switches

Table 21: Reference Search Modes

The drawings in the following subsections show how each reference search mode works. A linear stage with two end points and a moving slider is used as example.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

7.1.1 Mode 1

Reference search mode 1 only searches the left end switch. Select this mode by setting axis parameter #193 to 1. Figure 3 illustrates this.

Add 64 to the mode number (i.e. set axis parameter #193 to 65) to search the right end switch instead of the left end switch.

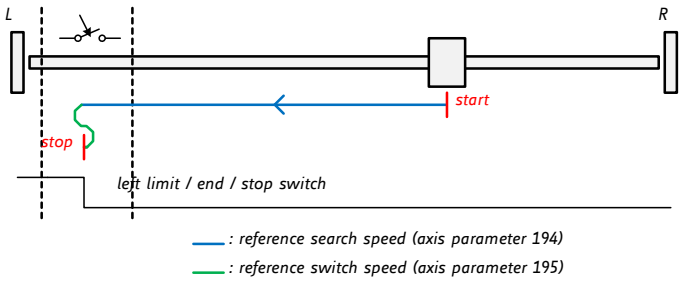


Figure 3: Reference Search Mode 1

7.1.2 Mode 2

Reference search mode 2 first searches the right end switch and then the left end switch. The left end switch is then used as the zero point. Figure 4 illustrates this. Select this mode by setting axis parameter #193 to 2. After the reference search has finished, axis parameter #196 contains the distance between the two reference switches in microsteps.

Add 64 to the mode number (i.e. set axis parameter #193 to 66) to search the left end switch first and then use the right end switch as the zero point.

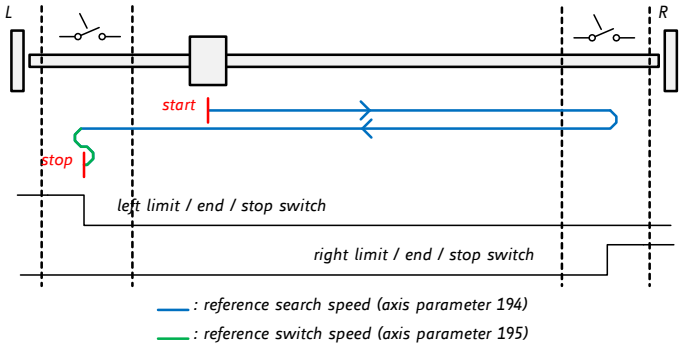


Figure 4: Reference Search Mode 2

7.1.3 Mode 3

Reference search mode 3 first searches the right end switch and then the left end switch. The left end switch is then searched from both sides, to find the middle of the left end switch. This is then used as the zero point. Figure 5 illustrates this. Select this mode by setting axis parameter #193 to 3. After the reference search has finished, axis parameter #196 contains the distance between the right end switch and the middle of the left end switch in microsteps.

Add 64 to the mode number (i.e. set axis parameter #193 to 67) to search the left end switch first and then use the middle of the right end switch as the zero point.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

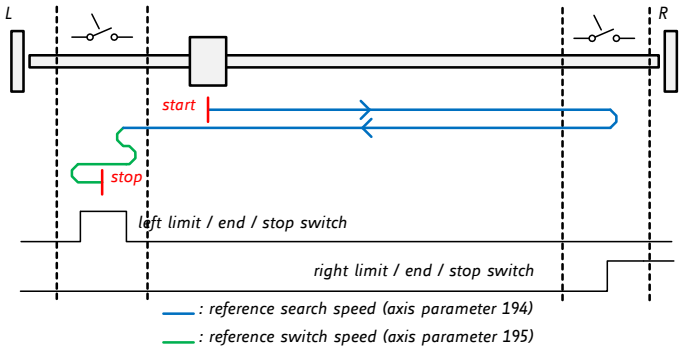


Figure 5: Reference Search Mode 3

7.1.4 Mode 4

Reference search mode 4 searches the left end switch only, but from both sides so that the middle of the switch will be found and used as the zero point. This is shown in figure 6.

Add 64 to the mode number (i.e. set axis parameter #193 to 68) to search the right end switch instead.

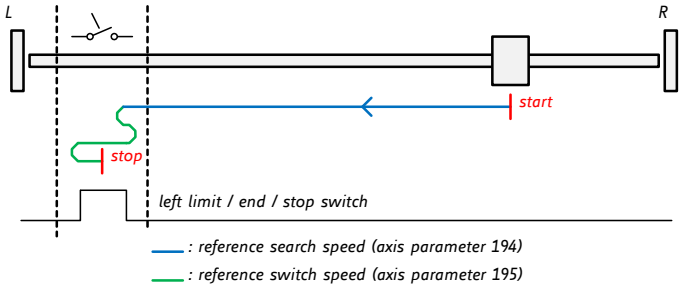


Figure 6: Reference Search Mode 4

7.1.5 Mode 5

Reference search mode 5 searches the home switch in negative direction. The search direction will be reversed if the left limit switch is reached. This is shown in figure 7.
Add 128 to the mode number (i.e. set axis parameter #193 to 133) to reverse the polarity of the home switch input.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

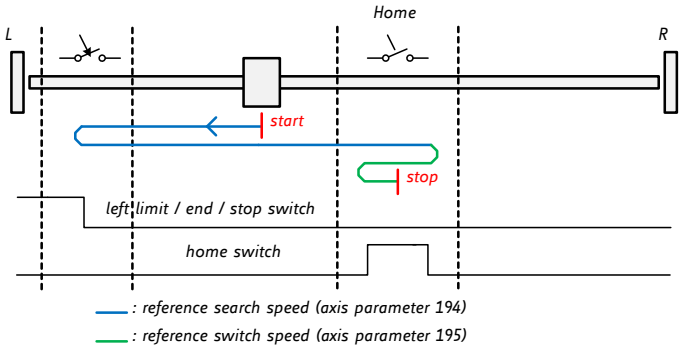


Figure 7: Reference Search Mode 5

7.1.6 Mode 6

Reference search mode 6 searches the home switch in positive direction. The search direction will be reversed if the right limit switch is reached. This is shown in figure 8.
Add 128 to the mode number (i.e. set axis parameter #193 to 134) to reverse the polarity of the home switch input.

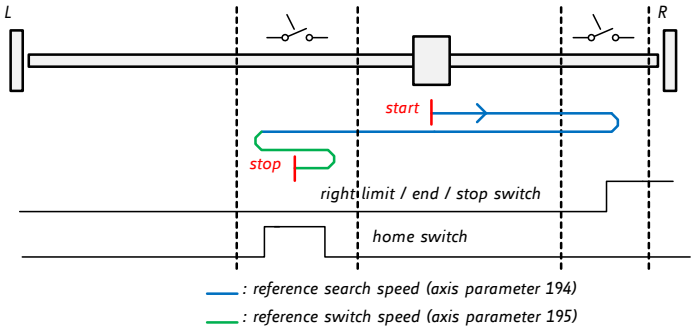


Figure 8: Reference Search Mode 6

7.1.7 Mode 7

Reference search mode 7 searches the home switch in negative direction, ignoring the limit switch inputs. It is recommended mainly for use with a circular axis. The exact middle of the switch will be found and used as the zero point. Figure 9 illustrates this.
Add 128 to the mode number (i.e. set axis parameter #193 to 135) to reverse the polarity of the home switch input.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

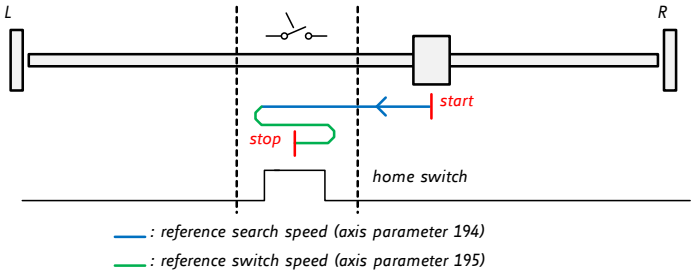


Figure 9: Reference Search Mode 7

7.1.8 Mode 8

Reference search mode 8 searches the home switch in positive direction, ignoring the limit switch inputs. It is recommended mainly for use with a circular axis. The exact middle of the switch will be found and used as the zero point. Figure 10 illustrates this.
Add 128 to the mode number (i.e. set axis parameter #193 to 136) to reverse the polarity of the home switch input.

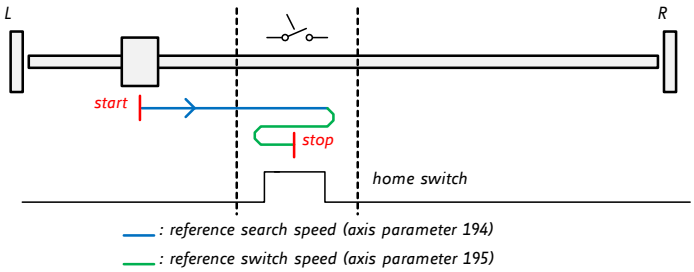


Figure 10: Reference Search Mode 8

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

7.2 StallGuard2

The module is equipped with motor driver chips that feature load measurement. This load measurement 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 StallGuard 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 automatically when the load gets too high. This function is configured mainly using axis parameters #174 and #181.

Stall detection can for example be used for finding the reference point without the need for reference switches. A short routine written in TMCL is needed to use StallGuard for reference searching.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

7.3 CoolStep

This section gives an overview of the CoolStep related parameters. Please bear in mind that the figure only shows one example for a drive. There are parameters which concern the configuration of the current. Other parameters are there for velocity regulation and for time adjustment.

Figure 11 shows all the adjustment points for CoolStep. It is necessary to identify and configure the thresholds for current (I6, I7 and I183) and velocity (V182). Furthermore the StallGuard2 feature has to be adjusted (SG170). It can also be enabled if needed (SG181).

The reduction or increasing of the current in the CoolStep area (depending on the load) has to be configured using parameters I169 and I171.

In this chapter only basic axis parameters are mentioned which concern CoolStep and StallGuard2. The complete list of axis parameters in chapter 4 contains further parameters which offer more configuration options.

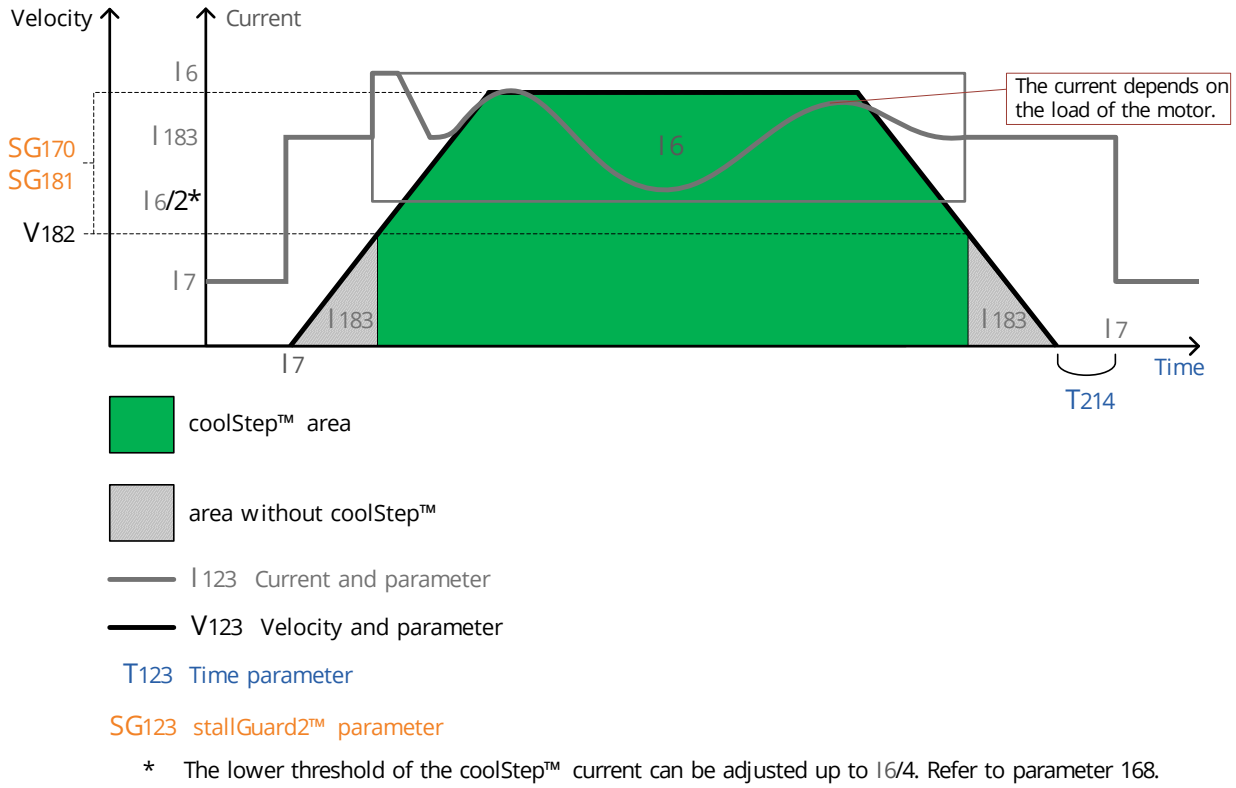


Figure 11: CoolStep Adjustment Points and Thresholds

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

CoolStep Adjustment Points and Thresholds		
Number	Axis Parameter	Description
I6	Absolute maximum current	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…255 divided by eight; e.g. step 0 = 0…7, step 1 = 8…15 and so on). Too high values may cause motor damage!
I7	Standby current	The current limit two seconds after the motor has stopped.
I168	smartEnergy current minimum	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
I169	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
I171	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
SG170	smartEnergy hysteresis	Sets the distance between the lower and the upper threshold for StallGuard2 reading. Above the upper threshold the motor current becomes decreased.
SG181	Stop on stall	Below this speed motor will not be stopped. Above this speed motor will stop in case StallGuard2 load value reaches zero.
V182	smartEnergy threshold speed	Above this speed CoolStep becomes enabled.
T214	Power down delay	Standstill period before the current is changed down to standby current. The standard value is 200 (which means 2000msec).

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Number	Axis Parameter	Description

Table 22: CoolStep Adjustment Points and Thresholds

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

7.4 Secondary Serial Address

Using the secondary serial address (global parameter #87) is an easy to use method to synchonize multiple TMCL modules controlled by one RS485 master. To use this method, set the secondary serial address of all modules connected to the bus to the same value (but not to 0 or to the value used for global parameter #66). All the serial addresses (global parameter #66) must be set to different values. Each command sent using the secondary address will then be executed by all modules at the same time, and there will be no bus contention as no replies are being sent. Using the normal serial address each module can still be addressed separately and will also still send back a reply.

7.5 Secondary CAN Address

Using the secondary CAN address (global parameter #83) is an easy to use method to synchonize multiple TMCL modules controlled by one CAN master. To use this method, set the secondary CAN address of all modules connected to the bus to the same value (but not to 0 or to the value used for global parameter #71). All the serial addresses (global parameter #71) must be set to different values. Each command sent using the secondary address will then be executed by all modules at the same time, and there will be no bus contention as no replies are being sent. Using the normal CAN address each module can still be addressed separately and will also still send back a reply.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

8 TMCL Programming Techniques and Structure

8.1 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.

8.2 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) stand alone TMCL programs look like this:

```
  //Initialization
  SAP 4, 0, 50000 //define maximum positioning speed
  SAP 5, 0, 10000 //define maximum 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
```

8.3 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. Please consider the following example:

```
//Define some constants
#include TMCLParam.tmc
MaxSpeed = 50000
MaxAcc = 10000
Position0 = 0
Position1 = 500000

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

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

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

```
  WAIT POS, Motor0, 0
17 JA MainLoop
```

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.

8.4 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 as well as STGP and RSGP 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.
- The STGP command stores the contents of a user varaible in the EEPROM.
- The RSGP command copies the value stored in the EEPROM back to the user variable.
- Global parameter 85 controls if user variables will be restored from the EEPROM automatically on startup (default setting) or not (user variables will then be initialized with 0 instead).

Please see the following example:

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

5 SGP MyVariable, 2, 1234   //Initialize the variable with the value 1234
  ...
7 ...
  GGP MyVariable, 2    //Copy contents of variable to accumulator register
9 CALC MUL, 2          //Multiply accumulator register with two
  AGP MyVariable, 2    //Store contents of accumulator register to variable
11 ...
  ...
```

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.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

8.5 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.

8.6 Combining Direct Mode and Standalone Mode

Direct mode and standalone mode can also be combined. 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
```

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

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.

8.7 Make the TMCL Program start automatically

For stand-alone operation the module has to start the TMCL program in its memory automatically after power-on. In order to achieve this, switch on the Autostart option of the module. This is controlled by global parameter #77. There are different ways to switch on the Autostart option:

- Execute the command SGP 77, 0, 1 in direct mode (using the Direct Mode tool in the TMCL-IDE).
- Use the Global Parameters tool in the TMCL-IDE to set global parameter #77 to 1.
- Use the Autostart entry in the TMCL menu of the TMCL Creator in the TMCL-IDE. Go to the Autostart entry in the TMCL menu and select '"On"'.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

9 Figures Index

1 StallGuard2 Load Measurement as a Function of Load 6
2 Energy Efficiency Example with CoolStep 6
3 Reference Search Mode 1 127
4 Reference Search Mode 2 127
5 Reference Search Mode 3 128
6 Reference Search Mode 4 128
7 Reference Search Mode 5 129
8 Reference Search Mode 6 129
9 Reference Search Mode 7 130
10 Reference Search Mode 8 130
11 CoolStep Adjustment Points and Thresholds 132

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

10 Tables Index

1 Most important Axis Parameters . . . 8
2 TMCL Command Format 11
3 TMCL Reply Format 12
4 TMCL Status Codes 12
5 Overview of all TMCL Commands . . . 18
6 Motion Commands 18
7 Parameter Commands 19
8 Branch Commands 19
9 I/O Port Commands 20
10 Calculation Commands 20
11 Interrupt Processing Commands . . . 21
12 Interrupt Vectors 22
13 New TMCL Commands 24
14 TMCL Control Commands 109
15 Meaning of the Letters in the Access Column 110
16 All Axis Parameters of the TMCM-6110 Module 117
17 Meaning of the Letters in the Access Column 118
18 All Global Parameters of the TMCM-6110 Module in Bank 0 121
19 User Variables in Bank 2 122
20 Interrupt Parameters in Bank 3 123
21 Reference Search Modes 126
22 CoolStep Adjustment Points and Thresholds 134
23 Firmware Revision 145
24 Document Revision 145

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

11 Supplemental Directives

11.1 Producer Information

11.2 Copyright

TRINAMIC owns the content of this user manual in its entirety, including but not limited to pictures, logos, trademarks, and resources. © Copyright 2022 TRINAMIC. All rights reserved. Electronically published by TRINAMIC, Germany.

Redistribution of sources or derived formats (for example, Portable Document Format or Hypertext Markup Language) must retain the above copyright notice, and the complete data sheet, user manual, and documentation of this product including associated application notes; and a reference to other available product-related documentation.

11.3 Trademark Designations and Symbols

Trademark designations and symbols used in this documentation indicate that a product or feature is owned and registered as trademark and/or patent either by TRINAMIC or by other manufacturers, whose products are used or referred to in combination with TRINAMIC's products and TRINAMIC's product documentation.

This TMCL™ Firmware Manual is a non-commercial publication that seeks to provide concise scientific and technical user information to the target user. Thus, trademark designations and symbols are only entered in the Short Spec of this document that introduces the product at a quick glance. The trademark designation /symbol is also entered when the product or feature name occurs for the first time in the document. All trademarks and brand names used are property of their respective owners.

11.4 Target User

The documentation provided here, is for programmers and engineers only, who are equipped with the necessary skills and have been trained to work with this type of product.

The Target User knows how to responsibly make use of this product without causing harm to himself or others, and without causing damage to systems or devices, in which the user incorporates the product.

11.5 Disclaimer: Life Support Systems

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.

Information given in this document is believed to be accurate and reliable. However, no responsibility is assumed for the consequences of its use nor for any infringement of patents or other rights of third parties which may result from its use. Specifications are subject to change without notice.

11.6 Disclaimer: Intended Use

The data specified in this user manual is intended solely for the purpose of product description. No representations or warranties, either express or implied, of merchantability, fitness for a particular purpose

or of any other nature are made hereunder with respect to information/specification or the products to which information refers and no guarantee with respect to compliance to the intended use is given.

In particular, this also applies to the stated possible applications or areas of applications of the product. TRINAMIC products are not designed for and must not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (safety-Critical Applications) without TRINAMIC's specific written consent.

TRINAMIC products are not designed nor intended for use in military or aerospace applications or environments or in automotive applications unless specifically designated for such use by TRINAMIC. TRINAMIC conveys no patent, copyright, mask work right or other trade mark right to this product. TRINAMIC assumes no liability for any patent and/or other trade mark rights of a third party resulting from processing or handling of the product and/or any other use of the product.

11.7 Collateral Documents & Tools

This product documentation is related and/or associated with additional tool kits, firmware and other items, as provided on the product page at: www.trinamic.com.

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

12 Revision History

12.1 Firmware Revision

Version	Date	Author	Description
1.10	2011-AUG-22	OK	First version supporting all features
1.11	2011-SEP-05	OK	WAIT command updated. Inverting home switch during reference search corrected.
1.12	2011-DEC-20	OK	STAP/RSAP 140 corrected.
1.13	2012-JAN-17	OK	Adjustments related to hardware version 1.2. Full functionality preserved with hardware version 1.1. USB bus powered operation: abandonment of TMCL-program (resp. no automatic start) in case of undervoltage. If the voltage is sufficient: re-initialization resp. auto start. New axis parameter #200 (boost current).
1.14	2012-FEB-29	OK	coolStep updated. Positioning range enlarged: 32 bit. Pin change interrupts for additional I/Os added. Heartbeat function added.
1.15	2012-MAR-12	OK	Global parameter #133 (random number) added.
1.16	2012-MAR-28	OK	Global parameter #84 (coordinate storage) added.
1.17	2012-APR-26	OK	Automatic current reduction (axis 3-5) updated.
1.18	2012-JUN-18	OK	GIO 255,2 and SIO 255, 2, x implemented.
1.19	2012-JUN-26	OK	Reference switch inputs can be inverted.
1.20	2012-OKT-04	OK	Global parameter #87 (secondary address for RS232/RS485) added. Reference search: the last position before setting the counter to zero can be read out with axis parameter #197.
1.21	2012-NOV-16	OK	Axis parameter #130 (minimum speed) updated.
1.22	2013-FEB-20	OK	Axis parameter #193 updated.
1.23…1.28	2014-FEB-28	OK	Not deployed.
1.29…1.30	2014-APR-23	OK	Not deployed.
1.31	2014-APR-23	OK	Improved command request target position reached.
1.34	2014-OCT-10	OK	RFS STATUS also copies result into accumulator.
1.35	2015-APR-8	OK	Using new USB IDs.
1.36…1.38	2015-MAY-24	OK	Not deployed.
1.39…1.41	2016-NOV-28	OK	Not deployed.
1.42	2017-FEB-14	OK	MVP COORD command corrected. New axis parameter #150.
1.43… 1.44		OK	Not deployed.
1.45	2018-AUG-10	OK	New TMCL commands implemented (not deployed).

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com

Version	Date	Author	Description
1.46	2019-DEC-16	OK	New RORA command corrected.
1.47	2021-OCT-07	OK	New commands ROLXA and RORXA fixed. Fixed: position reached message via USB sometimes got lost.

Table 23: Firmware Revision

12.2 Document Revision

Version	Date	Author	Description
V1.11	2017-FEB-14	OK	Document completely revised.
V1.12	2017-FEB-22	OK	Missing axis parameters #175…#179, #183 added.
V1.13	2017-MAR-13	OK	Superfluous interrupt numbers deleted.
V1.14	2017-APR-20	OK	Missing global parameter #90 added.
V1.15	2017-JUN-06	OK	Missing pull-up resistor switching added.
V1.17	2017-AUG-10	OK	Description of axis parameters #5 and #130 improved.
V1.18	2018-JAN-09	OK	Typo in description of axis parameter #204 corrected.
V1.19	2018-AUG-10	OK	Covers firmware revision V1.45 (not deployed).
V1.20	2020-MAY-27	OK	Superfluous entries in analog inputs table removed. Corrected information about switchable pull-up resistors. Covers firmware revision V1.46 (new TMCL commands).
V1.21	2021-SEP-14	OK	Description of MVP COORD command improved.
V1.22	2021-SEP-28	OK	New block diagram.
V1.23	2021-NOV-02	OK	New features of firmware V1.47.

Table 24: Document Revision

©2022 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
Terms of delivery and rights to technical change reserved.
Download newest version at www.trinamic.com