Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

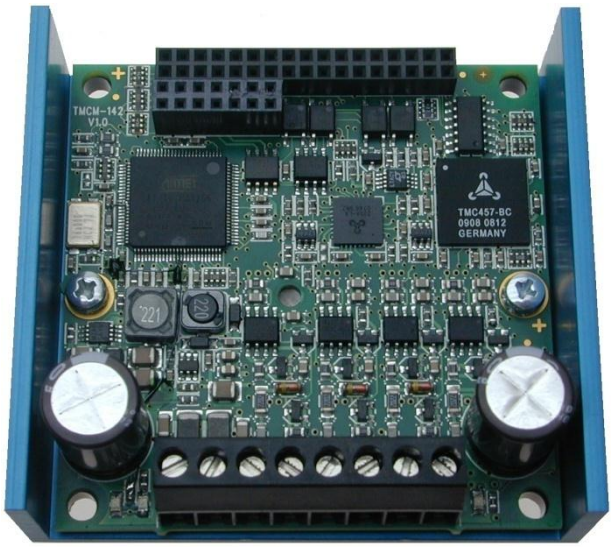
With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

TMCM-142



TMCL™ Firmware Manual

Version: 1.06
2014-JUN-24

TRINAMIC
MOTION CONTROL

Trinamic Motion Control GmbH & Co KG
Waterloohain 5
D - 22769 Hamburg, Germany
http://www.trinamic.com

Table of contents

1 Life support policy

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

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

2 Features

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

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

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

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

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

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

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

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

3 Order codes

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

Order code	Description	Dimensions [mm³]
TMCM-142-IF	Single axis stepper motor controller/driver, 5A RMS, 75V, with encoder feedback and the standard adapter/interface board TMCM-IF	76x70x33
Related motors		
QSH-5718	57mm/NEMA23, 1.8° step angle	57.2 x 57.2 x 41/55/ 78.5 mm
QSH-6018	60mm/NEMA24, 1.8° step angle	60.5 x 60.5 x 45/56/ 65/86 mm

Table 3.1: Order codes

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

4 Overview

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

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

5 Putting the TMCM-142 into operation

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

The things you need:

- TMCM-142-IF, consisting of TMCM-142 base and standard TMCM-IF adapter/interface add-on board.
- Interface (RS232, RS485, USB or CAN) suitable to your TMCM-142-IF with cables
- Nominal supply voltage +24V DC (+18...+78.5V DC) for your module
- A stepper motor which fit to your module, for example QSH-5718 or QSH-6018.
- TMCL-IDE program and PC
- Encoder optional

Precautions:

- ***Do not connect or disconnect the TMCM-142 and the TMCM-IF while powered!***
- ***Do not connect or disconnect the motor while powered!***
- Do not mix up connections or short-circuit pins.
- Avoid bounding I/O wires with motor power wires as this may cause noise picked up from the motor supply.
- Do not exceed the maximum power supply of 78.5V DC.
- ***Start with power supply OFF!***

5.1 Starting up

1. **Connect the TMCM-142 and the TMCM-IF**
   Usually TRINAMIC delivers the base board and the add-on board connected. If not for any reason, this figure will show you how to do this.

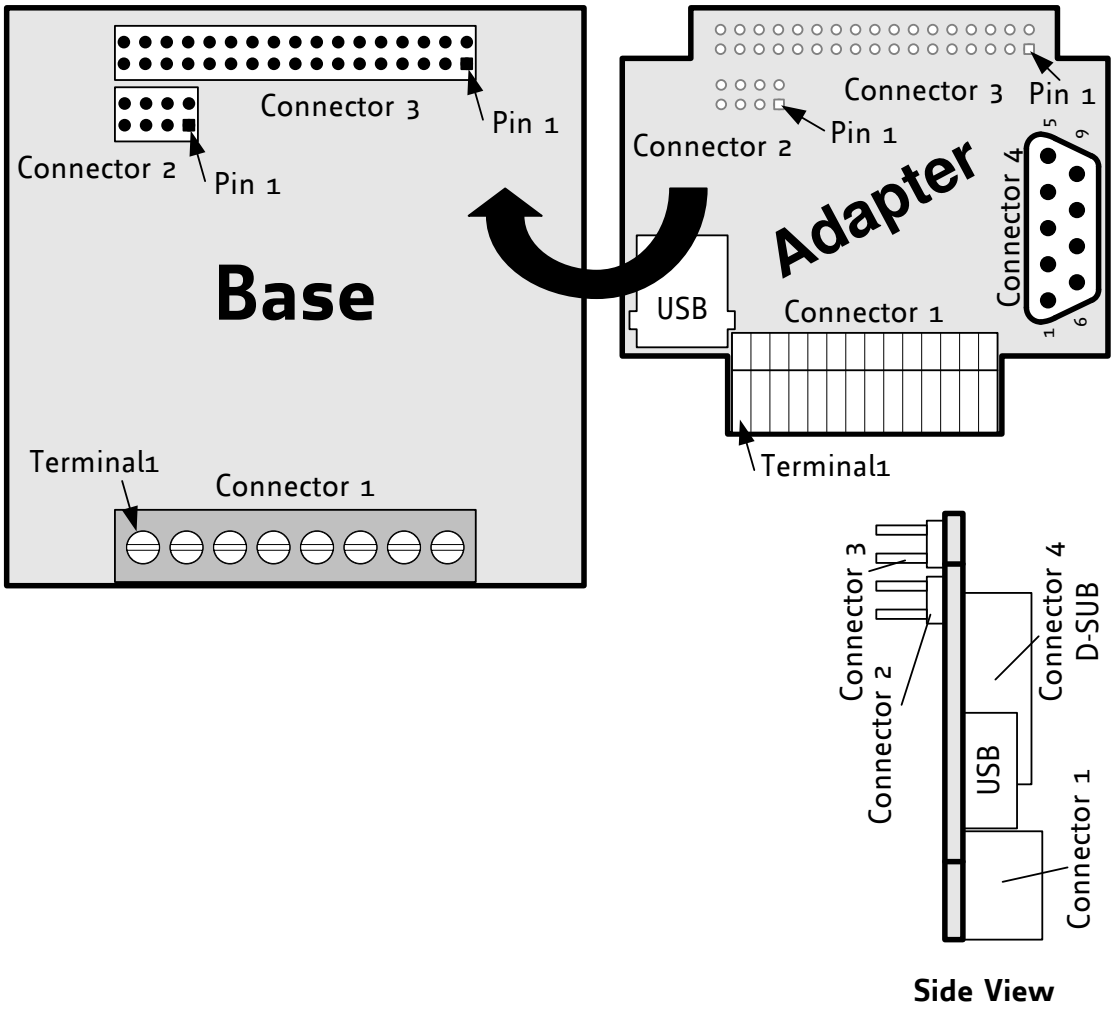


Figure 5.1: Connectors of the TMCM-142 and the TMCM-IF

2. **Connect the motor and the power supply**

Connect the motor and the power supply with ***connector 1*** of the TMCM-142:

Pin	Label	Description
1	NC	Not connected
2	NC	Not connected
3	GND	Supply ground
4	+V	Supply Voltage
5	B 2	Motor connection, Coil B
6	B 1	Motor connection, Coil B
7	A 2	Motor connection, Coil A
8	A 1	Motor connection, Coil A

Table 5.1: Base connector 1 - 1x8 pin, 5mm pitch screw connector

Attention: Do not exceed the maximum power supply of 78.5V DC.

3. Connect the interface

In this case we choose the USB interface for serial communication. USB is one out of five different interfaces available for communication with the TMCM-142-IF. You can refer to the hardware manuals of the TMCM-142 and the TMCM-IF for further information about the pinning of other interfaces.

Connect the USB interface:

Choose the USB port of the TMCM-IF and connect the interface with a USB cable. Accordingly, adjust the DIP switches.

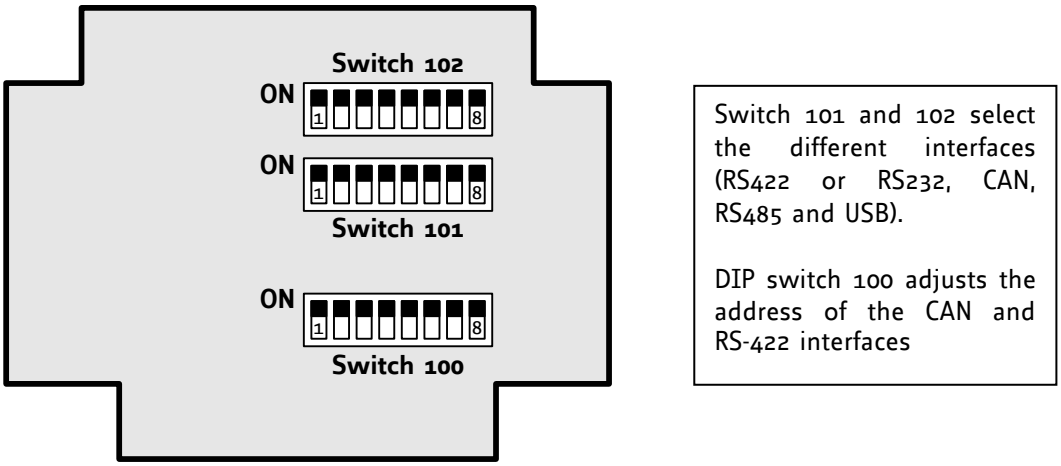


Figure 5.2: Overview of DIP switches

For selecting the USB interface, configure the DIP switches 101 and 102 as shown below:

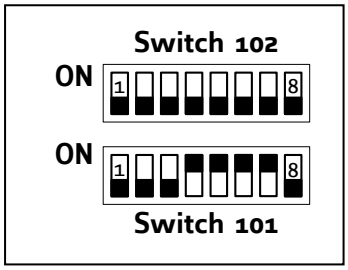


Figure 5.3: Configuration of DIP switches for USB

4. Connect the encoder

Differential and single ended incremental encoders with/without zero/index channel are supported.

If you want to use an encoder to meet your needs, you can connect as follows:

- Single ended encoder: GND to pin 20
  +5V to pin 16
  A to pin 34
  N to pin 29
  B to pin 33

- Differential encoder: GND to pin 20
  +5V to pin 16
  A+ to pin 34, A- to pin 12
  N+ to pin 29, N- to pin 18
  B+ to pin 33, B- to pin 14

Pin	Name	Function	PIN	Name	Function
1	TX-	RS422 Transmit – (data out from indexer)	2	TX+	RS422 Transmit + (data out from indexer)
3	RX-	RS422 Receive – (data into indexer)	4	RX+	RS422 Receive + (data out from indexer)
5		Internally pulled down via 2k7 resistor. Not supported by TMCL™.	6	IN0_A/D	Analog user controlled input #0. No internal resistors.
7	REF R	Optically isolated, active low limit switch input *Right*	8	STEP_ OUT/RXD	Step clock output from indexer RS232 option: RS232 receive
9	OUT_1	User controlled output #1. No internal resistors.	10	DIR_OUT/TXD	Direction output from indexer. RS232 option: RS232 transmit
11	IN7	Digital user controlled input #7. Optically isolated, active low (needs power supply on pin 15)	12	ENC_A-	Differential encoder: Channel A- input (optional)
13	IN2_A/D	Analog user controlled input #2. No internal resistors.	14	ENC_B-	Differential encoder: Channel B- input (optional)
15	+5V	DC bias for input opto couplers	16	+5VDC	Logic supply out for encoder
17	OUT_0	User controlled output #0. No internal resistors.	18	ENC_N-	Differential encoder: Channel N- input (optional)
19	REF L	Optically isolated, active low limit switch input *Left*	20	GND	Logic supply ground connection
21	IN3	Digital user controlled input #3. No internal resistors. (TTL)	22	OUT_2	User controlled output #2. No internal resistors.
23	IN8	Digital user controlled input #8. Optically isolated, active low (needs power supply on pin 15)	24		Not supported by TMCL™.
25	IN5	Digital user controlled input #5. No internal resistors. (TTL)	26	IN1_A/D	Analog user controlled input #1. No internal resistors.
27	ALARM	High voltage open collector output indicating driver fault condition.	28		Not supported by TMCL™.
29	ENC_N+	Encoder option: Single ended: Channel N input Differential: Channel N+ input	30	IN6	Digital user controlled input #6. No internal resistors. (TTL)
31	FS	Active for one clock pulse at each on-pole fullstep position.	32	IN4	Digital user controlled input #4. No internal resistors. (TTL)

Pin	Name	Function	PIN	Name	Function
33	ENC_B+	Encoder option: Single ended: Channel B input Differential: Channel B+ input	34	ENC_A+	Encoder option: Single ended: Channel A input Differential: Channel A+ input

Table 5.2: TMCM-IF connector 3 and TMCM-142 connector 3

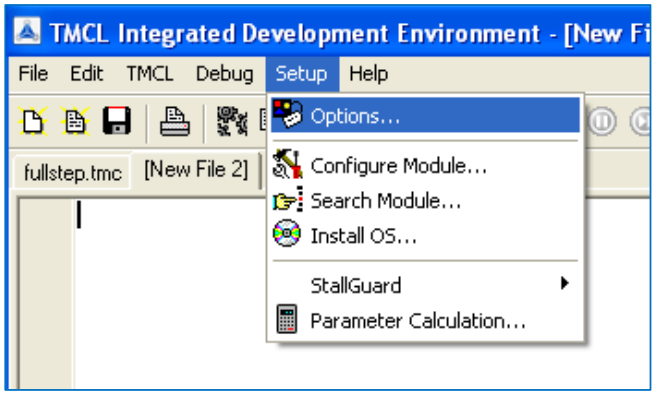
5. **Switch power supply *ON***
   The LED for power should glow now. This indicates that the on-board +5V supply is available.

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

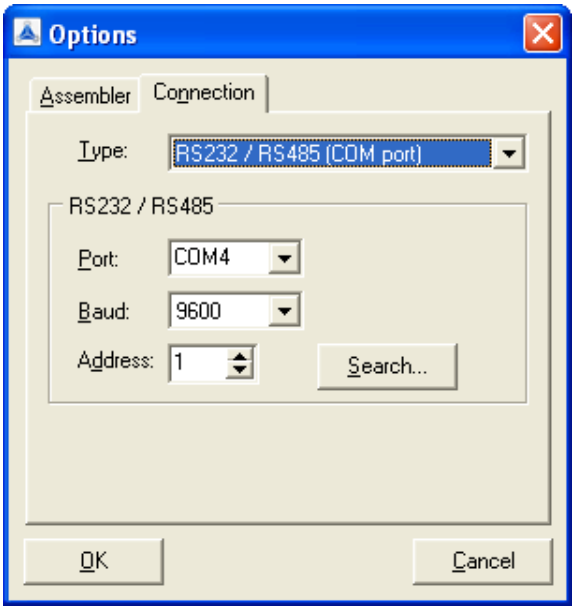
6. **Start the TMCL-IDE software development environment**
   The TMCL-IDE is on hand on the TechLibCD and on www.trinamic.com.

   Installing the TMCL-IDE:

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



   - Choose ***COM port*** and ***type*** with the parameters shown below (baud rate 9600). Click *OK*.

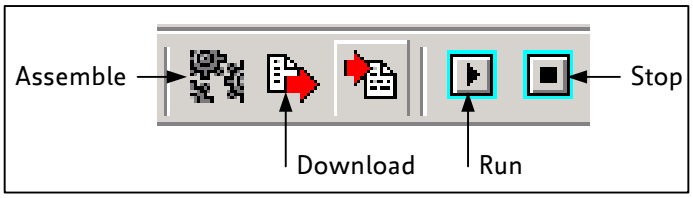


5.2 Testing with a simple TMCL™ program

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

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

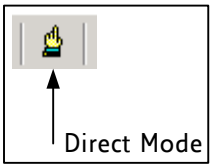
```
//A simple example for using TMCL™ and TMCL-IDE

        ROL 0, 500000                   //Rotate motor 0 with speed 500000
        WAIT TICKS, 0, 500
        MST 0
        ROR 0, 250000                   //Rotate motor 1 with 250000
        WAIT TICKS, 0, 500
        MST 0

        SAP 4, 0, 500000                //Set max. Velocity
        SAP 5, 0, 50000                 //Set max. Acceleration
Loop:   MVP ABS, 0, 10000000            //Move to Position 10000
        WAIT POS, 0, 0                  //Wait until position reached
        MVP ABS, 0, -1000000            //Move to Position -10000
        WAIT POS, 0, 0                  //Wait until position reached
        JA Loop                         //Infinite Loop
```



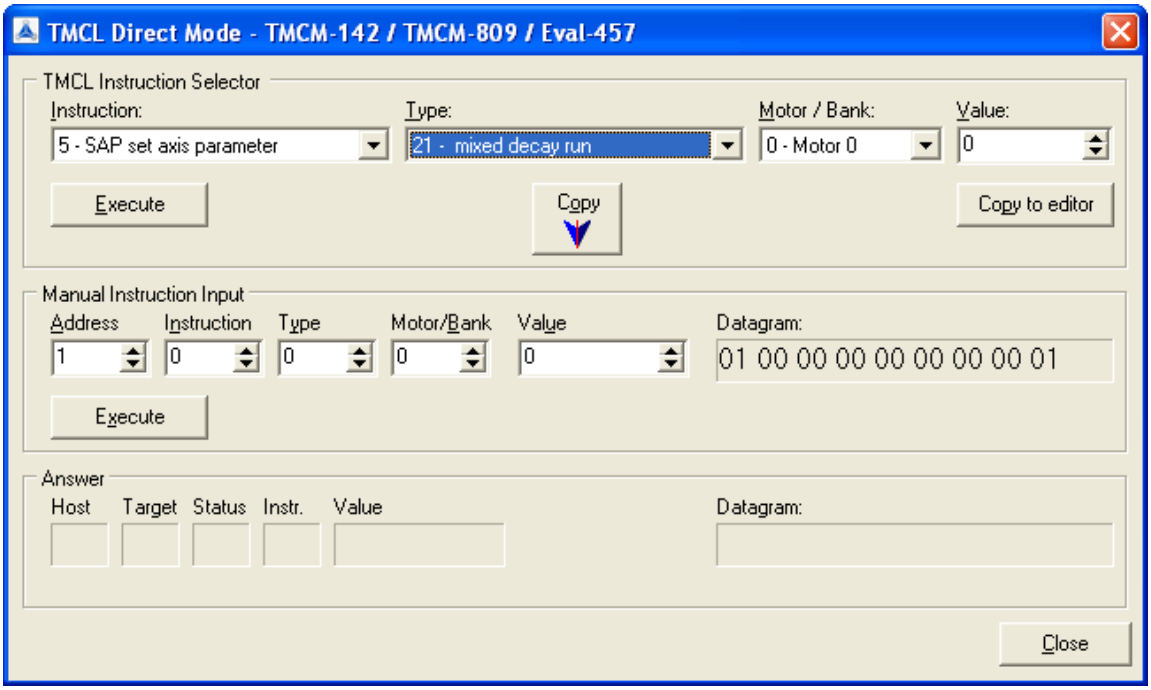
7. Click on Icon ***Assemble*** to convert the TMCL™ into machine code.
8. Then download the program to the TMCM-142 module via the icon ***Download***.
9. Press icon ***Run***. The desired program will be executed.
10. Click ***Stop*** button to stop the program.

5.3 Operating the module in direct mode

1. Start TMCL™ ***Direct Mode***.



2. If the communication is established the TMCM-142-IF is automatically detected. ***If the module is not detected, please check all points above (cables, interface, power supply, COM port, baud rate).***
3. Issue a command by choosing ***instruction***, ***type*** (if necessary), ***motor***, and ***value*** and click ***Execute*** to send it to the module.



Examples:

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

You will find a description of all TMCL™ commands in the following chapters.

6 TMCL™ and TMCL-IDE

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

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

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

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

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

6.1 Binary command format

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

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

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

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

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

Checksum calculation

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

- in C:

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

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

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

- in Delphi:

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

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

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

6.2 Reply format

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

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

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

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

6.2.1 Status codes

The reply contains a status code.

The status code can have one of the following values:

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

6.3 Stand-alone applications

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

6.4 TMCL™ command overview

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

6.4.1 Motion commands

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

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

6.4.2 Parameter commands

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

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

6.4.3 I/O port commands

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

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

6.4.4 Control commands

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

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

6.4.5 Calculation commands

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

Mnemonic	Command number	Meaning
CALC	19	Calculate using the accumulator and a constant value
CALCX	33	Calculate using the accumulator and the X register
AAP	34	Copy accumulator to an axis parameter
AGP	35	Copy accumulator to a global parameter
ACO	39	Copy accu to coordinate

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

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

6.5 TMCL™ commands

The following TMCL™ commands are currently supported:

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

Command	Number	Parameter	Description
GCO	31	<coordinate number>, <motor number>	Get coordinate
CCO	32	<coordinate number>, <motor number>	Capture coordinate
CALCX	33	<operation>	Process accumulator & X-register
AAP	34	<parameter>, <motor number>	Accumulator to axis parameter
AGP	35	<parameter>, <bank>	Accumulator to global parameter
ACO	39	<coordinate number, <motor number>	Accu to coordinate

TMCL™ control commands:

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

6.6 The ASCII interface

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

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

6.6.1 Format of the command line

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

Here are some examples for valid command lines:

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

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

6.6.2 Format of a reply

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

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

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

6.6.3 Commands that can be used in ASCII mode

The following commands can be used in ASCII mode: ROL, ROR, MST, MVP, SAP, GAP, STAP, RSAP, SGP, GGP, STGP, RSGP, RFS, SIO, GIO, SAC, SCO, GCO, CCO, UF0, UF1, UF2, UF3, UF4, UF5, UF6, and UF7.

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

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

6.6.4 Configuring the ASCII interface

The module can be configured so that it starts up either in binary mode or in ASCII mode. ***Global parameter 67 is used for this purpose*** (please see also chapter 8.1). Bit 0 determines the startup mode: If this bit is set, the module starts up in ASCII mode, else it will start up in binary mode (default). Bit 4 and Bit 5 determine how the characters that are entered are echoed back. Normally, both bits are set to zero. In this case every character that is entered is echoed back when the module is addressed. A Character can also

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

6.7 Commands

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

6.7.1 ROR (rotate right)

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

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

The module is based on the TMC457 motor controller and the TMC239 power driver. This makes possible choosing a velocity between 0 and 2147483647.

When axis parameter #255 (unit conversion mode) is set to 1 the speed must be given as microsteps per second. In this case the range for the speed is 0...31999999 microsteps/second.

Related commands: ROL, MST, SAP, GAP

Mnemonic: ROR 0, <velocity>

Binary representation:

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

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

Reply in direct mode:

STATUS	VALUE
100 – OK	(don't care)

Example:

Rotate right, motor #0, velocity = 350
Mnemonic: ROR 0, 350

Binary:

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	$01	$01	$00	$02	$00	$00	$01	$5e	$62

6.7.2 ROL (rotate left)

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

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

The module is based on the TMC457 motor controller and the TMC239 power driver. This makes possible choosing a velocity between 0 and 2147483647.

When axis parameter #255 (unit conversion mode) is set to 1 the speed must be given as microsteps per second. In this case the range for the speed is 0…31999999 microsteps/second.

Related commands: ROR, MST, SAP, GAP

Mnemonic: ROL 0, <velocity>

Binary representation:

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

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

Reply in direct mode:

STATUS	VALUE
100 – OK	(don't care)

Example:

Rotate left, motor #0, velocity = 1200
Mnemonic: ROL 0, 1200

Binary:

Byte Index	0	1	2	3	4	5	6	7	8
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0	Checksum
Value (hex)	$01	$02	$00	$00	$00	$00	$04	$b0	$b8