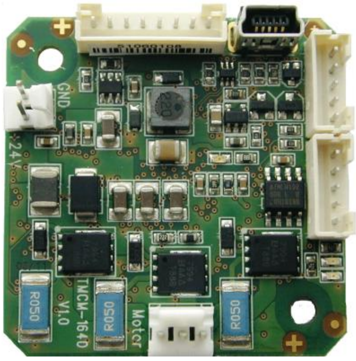
Firmware Version V1.47

TMCL™ FIRMWARE MANUAL



TMCM-1640

1-axis BLDC
controller / driver
5A / 24V DC
RS485 + USB interface
hall sensor interface
hall*FX*™
encoder interface

TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

www.trinamic.com

Table of contents

2 Features

The TMCM-1640 is a highly compact controller/driver module for brushless DC (BLDC) motors with up to 5A coil current, optional encoder and/or hall sensor feedback. For communication the module offers RS485 and (mini-)USB interface.

Applications

- Compact single-axis brushless DC motor solutions

Electrical data

- Supply voltage: +24VDC nom. (+12V… +28.5V DC)
- Motor current: up to 5A RMS (programmable)

Integrated motion controller

- High performance ARM Cortex™-M3 microcontroller for system control and communication protocol handling

Integrated driver

- High performance integrated pre-driver (TMC603)
- Support for sensorless back EMF commutation (hall*FX*™)
- High-efficient operation, low power dissipation (MOSFETs with low $R_{DS(ON)}$)
- Dynamic current control
- Integrated protection

Interfaces

- USB (mini-USB, full speed (12Mbit/s)) serial communication interfaces
- RS485 serial communication interface
- Hall sensor interface (+5V TTL or open-collector signals)
- Encoder interface (+5V TTL or open-collector signals)
- general purpose inputs (2x digital (+5V / +24V compatible), 1x analogue (+0… +10V)
- 1 general purpose output (open-drain)

Software

- Available with TMCL™
- Standalone operation or remote controlled operation
- Program memory (non volatile) for up to 2048 TMCL™ commands
- PC-based application development software TMCL-IDE available
- PC-based application development software TMCL-BLDC for initial adjustments

3 Overview

The software running on the microprocessor of the TMCM-1640 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 is related to the standard TMCL™ firmware [TMCL] with regard to protocol and commands. Corresponding, the module is based on the ARM Cortex-M3 microcontroller and the high performance pre-driver TMC603 and supports the standard TMCL™ with a special range of values.

4 Putting the TMCM-1640 into operation

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

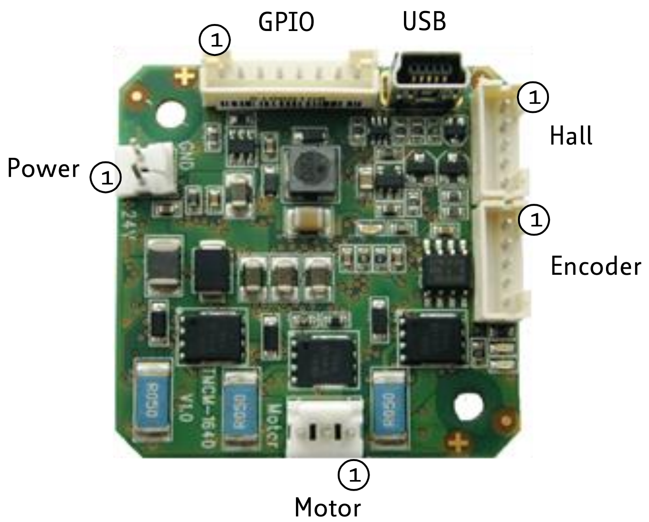
THINGS YOU NEED:

- TMCM-1640
- USB interface and appropriate cable or RS485 interface / adapter and appropriate cable
- Nominal supply voltage +24V DC (+12… +28.5V DC) for your module with sufficient output filtering (to be sure add e.g. 2200µF capacitor close to power supply input of module)
- BLDC motor, e.g. one of TRINAMICs QBL4208 motors
- Encoder optional
- TMCL-IDE program and PC

PRECAUTIONS
- Do not mix up connections or short-circuit pins.
- Avoid bounding I/O wires with motor power wires as this may cause noise picked up from the motor supply.
- The power supply has to be buffered by a capacitor. Otherwise the module will be damaged!
- Do not exceed the maximum power supply of 28.5V DC.
- Do not connect or disconnect the motor while powered!
- Start with power supply OFF!

4.1 Starting up

The following figure shows how the connectors have to be used.



Domain	Connector type	Mating connector type
Power	Tyco electronics (formerly AMP) MTA-100 series (3-640456-2), 2 pol., male	MTA 100 series (3-640441-2), 2 pol., female
Motor	Tyco electronics (formerly AMP) MTA-100 series (3-640456-3), 3 pol., male	MTA 100 series (3-640441-3), 3 pol., female
USB	5-pin standard mini-USB connector, female	5-pin standard mini-USB connector, male
Hall	2mm pitch 5 pin JST B5B-PH-K connector	Housing: JST PHR-5 Crimp contacts: BPH-002T-P0.5S (0.5-0.22mm)
Encoder	2mm pitch 5 pin JST B5B-PH-K connector	Housing: JST PHR-5 Crimp contacts: BPH-002T-P0.5S (0.5-0.22mm)
I/O, RS485	2mm pitch 8 pin JST B8B-PH-K connector	Housing: JST PHR-8 Crimp contacts: BPH-002T-P0.5S (0.5-0.22mm)

1. Connect the motor:

Pin	Label	Description
1	BM1	Motor coil phase 1 / U
2	BM2	Motor coil phase 2 / V
3	BM3	Motor coil phase 3 / W

2. Connect the encoder (optional):

Pin	Label	Description
1	GND	Encoder supply and signal ground
2	+5V	+5V output for encoder supply (max. 100mA)
3	A	Encoder channel a
4	B	Encoder channel b
5	N	Encoder index / null channel

3. Connect the hall sensor:

Pin	Label	Description
1	GND	Encoder supply and signal ground
2	+5V	+5V output for hall sensor supply
3	HALL_1	Hall sensor signal 1
4	HALL_2	Hall sensor signal 2
5	HALL_3	Hall sensor signal 3

4. Connect the I/Os and the RS485 interface, if needed:

Pin	Label	Description
1	GND	Signal and system ground
2	+5V	+5V output for supply of external circuit (max. 100mA)
3	AIN	Analog input (0..10V), may be used as velocity control input in standalone mode (depending on firmware)
4	IN_0	Digital input, may be used as stop (STOP_R) / limit switch input (depending on firmware)
5	IN_1	Digital input, may be used as stop (STOP_L) / limit switch input (depending on firmware)
6	OUT	Digital output (open-drain, max. 100mA)
7	RS485+	RS485 2-wire serial interface (non-inverted signal)
8	RS485-	RS485 2-wire serial interface (inverted signal)

5. Connect the USB interface:

Pin	Label	Description
1	VBUS	+5V power
2	D-	Data –
3	D+	Data +
4	ID	Not connected
5	GND	Ground

6. Connect the power supply as follows:

Pin	Label	Description
1	+U	Module and driver stage power supply input
2	GND	Module ground (power supply and signal ground)

Please note, that there is no protection against reverse polarity and only limited protection against voltages above the upper maximum limit. The power supply typically should be within a range of +12 to +28.5V.

When using supply voltages near the upper limit, a regulated power supply is mandatory. Please ensure that enough power filtering capacitors are available in the system (2200µF or more recommended) in order to absorb energy fed back by the motor while the motor is decelerating and in order to prevent any voltage surge e.g. during power-on (especially with longer power supply cables as there are only ceramic filter capacitors on-board). In larger systems an additional external zener/suppressor diode with adequate voltage rating might be necessary in order to limit the maximum voltage.

To ensure reliable operation of the unit, the power supply has to have a sufficient output capacitor and the supply cables should have a low resistance, so that the chopper operation does not lead to an increased power supply ripple directly at the unit. Power supply ripple due to the chopper operation should be kept at a maximum of a few 100mV

GUIDELINES FOR POWER SUPPLY:

a) keep power supply cables as short as possible
b) use cables with large diameters for power supply cables
c) add 2200µF or larger filter capacitors near the motor driver unit especially if the distance to the power supply is large (i.e. more than 2-3m)

7. Switch ON the power supply
The power LED glows now.
If this does not occur, switch power OFF and check your connections as well as the power supply.

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

Installing the TMCL-IDE
Make sure the COM port you intend to use is not blocked by another program.
Open TMCL-IDE by clicking ***TMCL.exe***.

Choose ***Setup*** and ***Options*** and thereafter the ***Connection tab***. Choose ***Type***. The TMCL-IDE shows you which ***Port*** the module uses. Click *OK*.

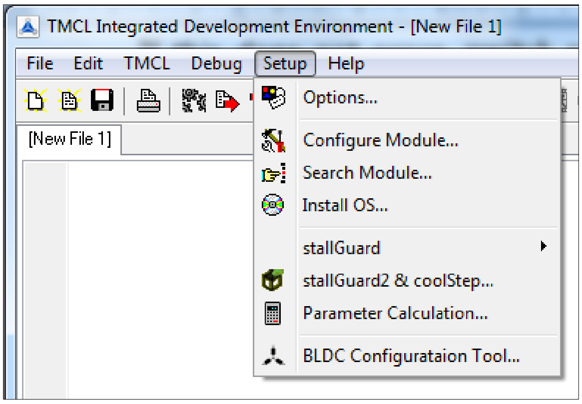


Figure 4.1: Setup menu

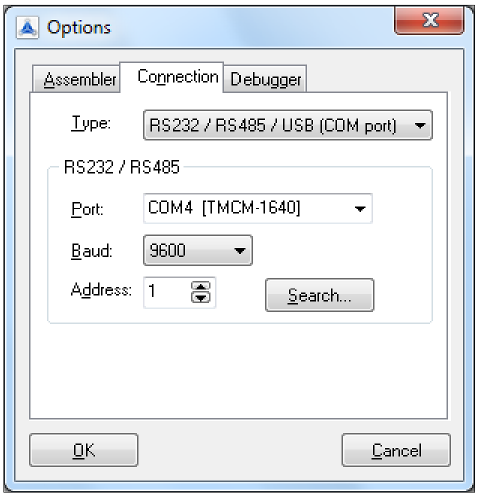
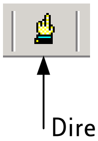


Figure 4.2: Connection tab of TMCL-IDE

4.2 Operating the module in direct mode

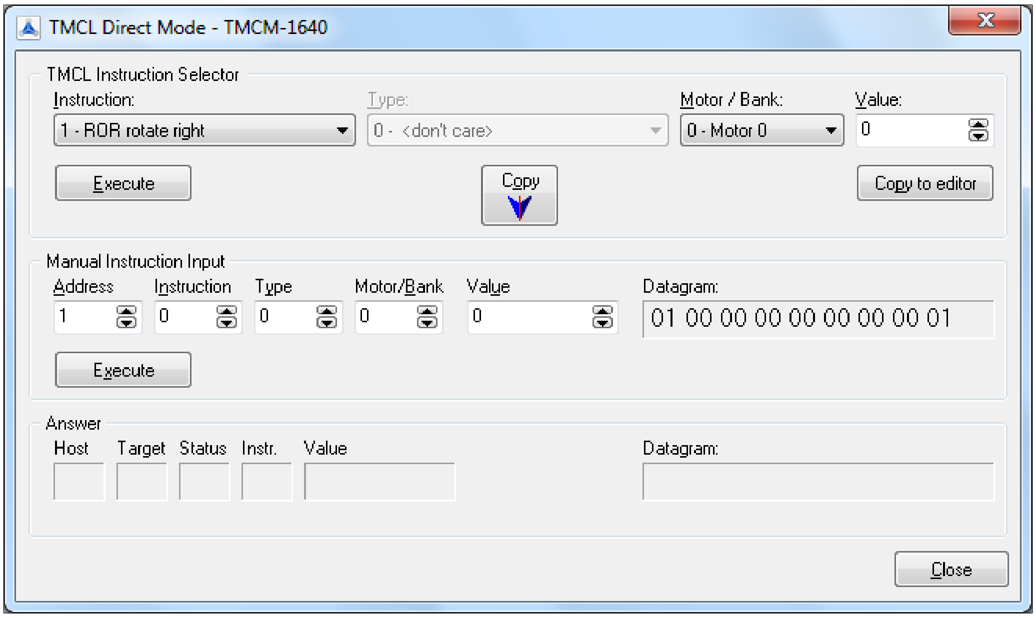
Start TMCL™ ***Direct Mode***.



Direct Mode

If the communication is established the TMCM-1640 is automatically detected. If the module is not detected, please check all points above (cables, interface, power supply, COM port, baud rate).

Issue a command by choosing ***Instruction***, ***Type*** (if necessary), ***Motor***, and ***Value*** and click ***Execute*** to send it to the module.



<u>Examples:</u>

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.

5 TMCL™

The TMCM-1640 module supports TMCL™ direct mode and standalone TMCL™ program execution. You can store up to 2048 TMCL™ instructions on it.

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

The Trinamic Motion Control Language (TMCL™) provides a set of structured motion control commands. Every motion control command can be given by a host computer or can be stored on the TMCM-1640 to form programs that run standalone on the module.

Every command has a binary representation and a mnemonic:

- The binary format is used to send commands from the host to a module in direct mode.
- The mnemonic format is used for easy usage of the commands when developing standalone TMCL™ applications with the TMCL-IDE (IDE means *Integrated Development Environment*).

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

5.1 Binary command format

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 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.

The binary command format for USB and RS485 is structured as follows:

Bytes	Meaning
1	Module address
1	Command number
1	Type number
1	Motor or Bank number
4	Value (MSB first!)
1	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 is an example for the 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 the command back to the module
```

5.2 Reply format

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

The reply format for USB and RS485 is structured as follows:

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

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

5.2.1 Status codes

The reply contains a status code.

The status code can have one of the following values:

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

5.3 Standalone applications

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

5.4 Testing with a simple TMCL™ program

Open the file test2.tmc of the TMCL-IDE. The following source code appears on the screen:

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

Loop:
  ROL 0, 4000                //rotate left with 4000 rev/min
  WAIT TICKS, 0, 2000
  ROR 0, 4000                //rotate right with 4000 rev/min
  WAIT TICKS, 0, 2000
  JA Loop
```

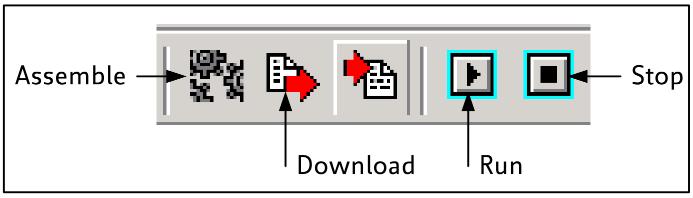


Figure 5.1: Assemble, download, stop, and run icons of TMCL-IDE

Click on icon ***Assemble*** to convert the TMCL™ example into binary code.
Then download the program to the TMCM-1640 module via the icon ***Download***.
Press icon ***Run***. The desired program will be executed.
Click ***Stop*** button to stop the program.

For further information about the TMCL-IDE and TMCL™ programming techniques please refer to the TMCL-IDE User Manual on TRINAMICs website.

TRINAMIC offers two software tools for BLDC applications: the *TMCL-BLDC* and the BLDC tool of the *TMCL-IDE*. Whereas the TMCL-BLDC is used for testing different configurations in all modes of operation the TMCL-IDE is mainly designed for conceiving programs and firmware updates. New versions of the TMCL-BLDC and the TMCL-IDE can be downloaded free of charge from the TRINAMIC website (http://www.trinamic.com).

5.5 TMCL™ command overview

The following section provides a short overview of the TMCL™ commands supported by the TMCM-1640.

5.5.1 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>	Motor stop movement
MVP	4	ABS|REL, <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)

Command	Number	Parameter	Description
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, e.g. communication settings or TMCL™ user variables)
GGP	10	<parameter>, <bank number>	Get global parameter (module specific, 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)
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
CALCX	33	<operation>	Process accumulator & X-register
AAP	34	<parameter>, <motor number>	Accumulator to axis parameter
AGP	35	<parameter>, <bank>	Accumulator to global parameter

5.5.2 Commands listed according to subject area

5.5.2.1 Motion commands

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

Mnemonic	Command number	Meaning
ROL	2	Rotate left
ROR	1	Rotate right
MVP	4	Move to position
MST	3	Motor stop

5.5.2.2 Parameter commands

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

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

5.5.2.3 Control commands

These commands are used to control the program flow (loops, conditions, jumps etc.) in standalone mode, only.

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

5.5.2.4 Calculation commands

These commands are intended to be used for calculations within TMCL™ applications in standalone mode, only. For calculating purposes there are an accumulator (or accu or A register) and an X register. When executed in a TMCL™ program (in standalone mode), all TMCL™ commands that read a value store the result in the accumulator. The X register can be used as an additional memory when doing calculations. It can be loaded from the accumulator.

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

Mixing standalone program execution and direct mode
It is possible to use some commands in direct mode while a standalone program is active. When a command which reads out a value is executed (direct mode) the accumulator will not be affected. While a TMCL™ program is running standalone on the module, a host can still send commands like GAP and GGP to it (e.g. to query the actual position of the motor) without affecting the flow of the TMCL™ program running standalone on the module.

5.6 Commands

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

5.6.1 ROR (rotate right)

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 #2 (*target velocity*).

Related commands: ROL, MST, SAP, GAP

Mnemonic: ROR 0, <velocity>

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE <velocity>
1	don't care	0	-2147483648… +2147483647

Reply in direct mode:

STATUS	COMMAND	VALUE
100 – OK	1	don't care

Example:
Rotate right, velocity = 350
Mnemonic: ROR 0, 350

Binary:

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

5.6.2 ROL (rotate left)

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 #2 (*target velocity*).

Related commands: ROR, MST, SAP, GAP

Mnemonic: ROL 0, <velocity>

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
2	don't care	0	<velocity> -2147483648… +2147483647

Reply in direct mode:

STATUS	COMMAND	VALUE
100 – OK	2	don't care

Example:

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

Binary:

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

5.6.3 MST (motor stop)

The motor will be instructed to stop.

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

Related commands: ROL, ROR, SAP, GAP

Mnemonic: MST 0

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
3	don't care	0	don't care

Reply in direct mode:

STATUS	COMMAND	VALUE
100 – OK	3	don't care

Example:

Stop motor
Mnemonic: MST 0

Binary:

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

5.6.4 MVP (move to position)

The motor will be instructed to move to a specified relative or absolute position. It uses the acceleration/deceleration ramp and the positioning speed programmed into the unit. This command is non-blocking (like all commands). A reply will be sent immediately after command interpretation and initialization of the motion controller. Further commands may follow without waiting for the motor reaching its end position. The maximum velocity and acceleration are defined by axis parameters #4 and #11.

Two operation types are available:

- Moving to an absolute position in the range from -2147483648… +2147483647.
- Starting a relative movement by means of an offset to the actual position. In this case, the new resulting position value must not exceed the above mentioned limits, too.

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

Related commands: SAP, GAP, and MST

Mnemonic: MVP <ABS|REL>, 0, <position|offset value>

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
4	0 ABS – absolute	0	<position> -2147483648… +2147483647
	1 REL – relative	0	<offset> -2147483648… +2147483647

Reply in direct mode:

STATUS	COMMAND	VALUE
100 – OK	4	don't care

Example MVP ABS:

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

Binary:

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$04	$00	$00	$00	$00	$23	$28

Example MVP REL:

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

Binary:

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

5.6.5 SAP (set axis parameter)

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

Related commands: GAP, STAP, and RSAP

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

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
5	<parameter number>	0	<value>

Reply in direct mode:

STATUS	COMMAND	VALUE
100 – OK	5	don't care

A list of all parameters which can be used for the SAP command is shown in section 6.

Example:

Set the absolute maximum current to 200mA
Mnemonic: SAP 6, 0, 200

Binary:

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

5.6.6 GAP (get axis parameter)

Most parameters of the TMCM-1640 can be adjusted individually. They can be read out using the GAP command.

Related commands: SAP, STAP, and RSAP

Mnemonic: GAP <parameter number>, 0

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
6	<parameter number>	0	don't care

Reply in direct mode:

STATUS	COMMAND	VALUE
100 – OK	6	don't care

A list of all parameters which can be used for the GAP command is shown in section 6.

Example:

Get the actual position motor position
Mnemonic: GAP 1, 0

Binary:

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

Reply:

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

5.6.7 STAP (store axis parameter)

The STAP command stores an axis parameter previously set with a *Set Axis Parameter command (SAP)* permanently. Most parameters are automatically restored after power up (refer to axis parameter list in chapter 6).

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

Related commands: SAP, RSAP, and GAP

Mnemonic: STAP <parameter number>, 0

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
7	<parameter number>	0	don't care*

*** The value operand of this function has no effect. Instead, the currently used value (e.g. selected by SAP) is saved.**

Reply in direct mode:

STATUS	COMMAND	VALUE
100 – OK	7	don't care

A list of all parameters which can be used for the STAP command is shown in section 6.

Example:

Store the maximum speed
Mnemonic: STAP 4, 0

Binary:

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

Note: The STAP command will not have any effect when the configuration EEPROM is locked. The error code 5 (configuration EEPROM locked) will be returned in this case.

5.6.8 RSAP (restore axis parameter)

For all configuration related axis parameters non-volatile memory locations are provided. By default, most parameters are automatically restored after power up (refer to axis parameter list in chapter 6). A single parameter that has been changed before can be reset by this instruction also.

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

Related commands: SAP, STAP, and GAP

Mnemonic: RSAP <parameter number>, 0

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
8	<parameter number>	0	don't care

Reply in direct mode:

STATUS	COMMAND	VALUE
100 – OK	8	don't care

A list of all parameters which can be used for the RSAP command is shown in section 6.

Example:

Restore the maximum current
Mnemonic: RSAP 6, 0

Binary:

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

5.6.9 SGP (set global parameter)

Global parameters are related to the host interface, peripherals or other application specific variables. The different groups of these parameters are organized in *banks* to allow a larger total number for future products. Currently, only bank 0 and 1 are used for global parameters, and only bank 2 is intended to use for user variables.

Related commands: GGP, STGP, RSGP

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

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
9	<parameter number>	<bank number>	<value>

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

A list of all parameters which can be used for the SGP command is shown in section 7.

Example: set variable 0 at bank 2 to 100
Mnemonic: SGP, 0, 2, 100

Binary:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$09	$00	$02	$00	$00	$00	$64

5.6.10 GGP (get global parameter)

All global parameters can be read with this function.

Related commands: SGP, STGP, RSGP

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

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
10	<parameter number>	<bank number>	don't care

Reply in direct mode:

STATUS	VALUE
100 – OK	<value>

A list of all parameters which can be used for the GGP command is shown in section 7.

Example: get variable 0 from bank 2
Mnemonic: GGP, 0, 2

Binary:

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

5.6.11 STGP (store global parameter)

Some global parameters are located in RAM memory, so modifications are lost at power down. This instruction copies a value from its RAM location to the configuration EEPROM and enables permanent storing. Most parameters are automatically restored after power up.

Related commands: SGP, GGP, RSGP

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

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
11	<parameter number>	<bank number>	don't care

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

A list of all parameters which can be used for the STGP command is shown in section 7.

Example: copy variable 0 at bank 2 to the configuration EEPROM
Mnemonic: STGP, 0, 2

Binary:

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

5.6.12 RSGP (restore global parameter)

This instruction copies a value from the configuration EEPROM to its RAM location and so recovers the permanently stored value of a RAM-located parameter. Most parameters are automatically restored after power up.

Related commands: SGP, GGP, STGP

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

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
12	<parameter number>	<bank number>	don't care

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

A list of all parameters which can be used for the RSGP command is shown in section 7.

Example: copy variable 0 at bank 2 from the configuration EEPROM to the RAM location
Mnemonic: RSGP, 0, 2

Binary:

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

5.6.13 CALC (calculate)

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

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

Mnemonic: CALC <op>, <value>

Binary representation:

COMMAND	TYPE <op>	MOT/BANK	VALUE
19	0 ADD – add to accu 1 SUB – subtract from accu 2 MUL – multiply accu by 3 DIV – divide accu by 4 MOD – modulo divide by 5 AND – logical and accu with 6 OR – logical or accu with 7 XOR – logical exor accu with 8 NOT – logical invert accu 9 LOAD – load operand to accu	don't care	<operand>

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

Binary:

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

5.6.14 COMP (compare)

The specified number is compared to the value in the accumulator register. The result of the comparison can be used for example by the conditional jump (JC) instruction. This command is intended for use in standalone operation, only. The host address and the reply are required to take the instruction to the TMCL™ program memory while the TMCL™ program downloads. It does not make sense to use this command in direct mode.

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

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

Mnemonic: COMP <value>

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
20	don't care	don't care	<comparison value>

Example:
Jump to the address given by the label when the position of the motor #2 is greater or equal to 1000.

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

Binary format of the COMP 1000 command:

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

5.6.15 JC (jump conditional)

The JC instruction enables a conditional jump to a fixed address in the TMCL™ program memory, if the specified condition is met. The conditions refer to the result of a preceding comparison. This function is for standalone operation only. The host address and the reply are required to take the instruction to the TMCL™ program memory while the TMCL™ program downloads. It is not possible to use this command in direct mode.

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

Related commands: JA, COMP, WAIT

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

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
21	0 ZE - zero 1 NZ - not zero 2 EQ - equal 3 NE - not equal 4 GT - greater 5 GE - greater/equal 6 LT - lower 7 LE - lower/equal 8 ETO - time out error 9 EAL - external alarm	don't care	<jump address>

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

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

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

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$15	$05	$00	$00	$00	$00	$0a

5.6.16 JA (jump always)

Jump to a fixed address in the TMCL™ program memory. This command is intended for standalone operation, only. The host address and the reply are required to take the instruction to the TMCL™ program memory while the TMCL™ program downloads. This command cannot be used in direct mode.

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

Related commands: JC, WAIT, CSUB

Mnemonic: JA <Label>

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
22	don't care	don't care	<jump address>

Example: An infinite loop in TMCL™

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

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

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

5.6.17 CSUB (call subroutine)

This function calls a subroutine in the TMCL™ program memory. It is intended for standalone operation, only. The host address and the reply are required to take the instruction to the TMCL™ program memory while the TMCL™ program downloads. This command cannot be used in direct mode.

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

Related commands: RSUB, JA

Mnemonic: CSUB <Label>

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
23	don't care	don't care	<subroutine address>

Example: Call a subroutine

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

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

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

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Instruction Number	Type	Motor/ Bank	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$01	$17	$00	$00	$00	$00	$00	$64

5.6.18 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.

The host address and the reply are only used to take the instruction to the TMCL™ program memory while the TMCL™ program loads down. This command cannot be used in direct mode.

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

Related command: CSUB

Mnemonic: RSUB

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
24	don't care	don't care	don't care

Example: RSUB

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

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

Binary format of RSUB:

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

5.6.19 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. The host address and the reply are only used to take the instruction to the TMCL™ program memory while the TMCL™ program downloads. This command is not to be used in direct mode.

There are different wait conditions that can be used:

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

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

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

Related commands: JC, CLE

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

Binary representation:

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

* One tick is 10msec (in standard firmware).

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

Binary:

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

5.6.20 STOP (stop TMCL™ program execution)

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

Every standalone TMCL™ program needs the STOP command at its end. It is not to be used in direct mode.

Internal function: TMCL™ instruction fetching is stopped.

Related commands: none
Mnemonic: STOP

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
28	don't care	don't care	don't care

Example:
Mnemonic: STOP

Binary:

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

5.6.21 CALCX (calculate using the X register)

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

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

Mnemonic: CALCX <operation>

Binary representation:

COMMAND	TYPE <operation>	MOT/BANK	VALUE
33	0 ADD – add X register to accu 1 SUB – subtract X register from accu 2 MUL – multiply accu by X register 3 DIV – divide accu by X-register 4 MOD – modulo divide accu by x-register 5 AND – logical and accu with X-register 6 OR – logical or accu with X-register 7 XOR – logical exor accu with X-register 8 NOT – logical invert X-register 9 LOAD – load accu to X-register 10 SWAP – swap accu with X-register	don't care	don't care

Example:
Multiply accu by X-register
Mnemonic: CALCX MUL
Binary:

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

5.6.22 AAP (accumulator to axis parameter)

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

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

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

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
34	<parameter number>	0	<don't care>

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

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

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

Binary format of the AAP 0,0 command:

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

5.6.23 AGP (accumulator to global parameter)

The content of the accumulator register is transferred to the specified global parameter. For practical use, the accumulator has to be loaded e.g. by a preceding GAP instruction. The accumulator may have been modified by the CALC or CALCX (calculate) instruction. ***Note that the global parameters in bank 0 are mostly EEPROM-only and thus should not be modified automatically by a standalone application.*** (See chapter 6 for a complete list of global parameters).

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

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

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
35	<parameter number>	<bank number>	don't care

Reply in direct mode:

STATUS	VALUE
100 – OK	don't care

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

Binary:

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

5.6.24 Customer specific TMCL™ command extension (UF0… UF7/user function)

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

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

Related commands: none

Mnemonic: UF0… UF7 <parameter number>

Binary representation:

COMMAND	TYPE	MOT/BANK	VALUE
64… 71	user defined	user defined	user defined

Reply in direct mode:

Byte Index	0	1	2	3	4	5	6	7
Function	Target-address	Target-address	Status	Instruction	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
Value (hex)	$02	$01	user defined	64… 71	user defined	user defined	user defined	user defined

5.6.25 TMCL™ Control Functions

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

Command	Type	Parameter	Description	Access
136	0 – string 1 – binary	Firmware version	Get the module type and firmware revision as a string or in binary format. (*Motor/Bank* and *Value* are ignored.)	read

Type set to 0 - reply as a string:

Byte index	Contents
1	Host Address
2… 9	Version string (8 characters, e.g. 1640V100)

There is no checksum in this reply format!

Type set to 1 - version number in binary format:
Please use the normal reply format. The version number is output in the *value* field.

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

6 Axis parameter overview (SAP, GAP, STAP, RSAP)

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

Meaning of the letters in column Access:

R = readable (GAP)
W = writable (SAP)
E = stored and automatically restored from EEPROM after reset or power-on

Number	Axis Parameter	Description	Range [Unit]	Access
0	Target position	The target position of a currently executed ramp.	-2147483648… +2147483647	RW
1	Actual position	Set/get the position counter without moving the motor.	-2147483648… +2147483647	RW
2	Target speed	Set/get the desired target velocity.	-2147483648… +2147483647 [rpm]	RW
3	Actual speed	The actual velocity of the motor.	-2147483648… +2147483647 [rpm]	R
4	Max. ramp velocity	The maximum velocity used for velocity ramp in velocity mode and positioning mode. Set this value to a realistic velocity which the motor can reach!	-2147483648… +2147483647 [rpm]	RWE
5	PWM limit	Set/get PWM limit (0%… 100%).	0… 3599	RWE
6	Max current	Set/get the max allowed motor current. *This value can be temporarily exceeded marginal due to the operation of the current regulator.	0… +4294967295 [mA]	RWE
7	MVP Target reached velocity	Maximum velocity at which end position can be set. Prevents issuing of end position when the target is passed at high velocity.	-2147483648… +2147483647 [rpm]	RWE
8	Threshold speed for velocity PID	Threshold speed for velocity regulation to switch between first and second velocity PID parameter set.	-2147483648… +2147483647 [rpm]	RWE
9	Motor halted velocity	If the actual speed is below this value the motor halted flag will be set.	-2147483648… +2147483647 [rpm]	RWE
10	MVP target reached distance	Maximum distance at which the position end flag is set.	-2147483648… +2147483647	RWE
11	Acceleration	Acceleration parameter for ROL, ROR, and the velocity ramp of MVP.	-2147483648… +2147483647 [RPM/s]	RWE
12	Threshold speed for position PID	Threshold speed for position regulation to switch between first and second position PID parameter set.	-2147483648… +2147483647 [rpm]	RWE
13	Ramp generator speed	The actual speed of the velocity ramp used for positioning and velocity mode.	-2147483648… +2147483647 [rpm]	R
14	velocity threshold for hallFX™	Velocity to switch from controlled to hallFX™ mode. Set this value to a realistic velocity which the motor can reach in controlled mode!	-2147483648… +2147483647 [rpm]	RWE
20	Switch 2 active	0: inactive 1: active (IO connector pin 4)	0/1	R
21	Switch 1 active	0: inactive 1: active (IO connector pin 5)	0/1	R
25	Thermal winding time constant	Thermal winding time constant for the used motor. Used for I^2t monitoring.	0… +4294967295 [ms]	RWE

Number	Axis Parameter	Description	Range [Unit]	Access
26	I^2t limit	An actual I^2t sum that exceeds this limit leads to increasing the I^2t exceed counter.	0… +4294967295	RWE
27	I^2t sum	Actual sum of the I^2t monitor.	0… +4294967295	R
28	I^2t exceed counter	Counts how often an I^2t sum was higher than the I^2t limit.	0… +4294967295	RWE
29	Clear I^2t exceeded flag	Clear the flag that indicates that the I^2t sum has exceeded the I^2t limit.	(ignored)	W
30	Minute counter	Counts the module operational time in minutes.	0… +4294967295 [min]	RWE
130	P parameter for position PID (I)	P parameter of position PID regulator (first parameter set)	-2147483648… +2147483647	RWE
131	I parameter for position PID (I)	I parameter of position PID regulator (first parameter set)	-2147483648… +2147483647	RWE
132	D parameter for position PID (I)	D parameter of position PID regulator (first parameter set)	-2147483648… +2147483647	RWE
133	PID regulation loop delay	PID calculation delay. Set PID operational frequency.	0… +4294967295 [ms]	RWE
134	Current regulation loop delay	Delay of current limitation algorithm / PID current regulator.	0… +4294967295 [50µs]	RWE
135	I-Clipping parameter for position PID (I)	I-Clipping parameter of position PID regulator (first parameter set) (A too high value causes overshooting at positioning mode.)	-2147483648… +2147483647	RWE
136	PWM-Hysteresis	Compensates dead time of PWM and motor friction.	0… +4294967295	RWE
140	P parameter for velocity PID (I)	P parameter of velocity PID regulator (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
141	I parameter for velocity PID (I)	I parameter of velocity PID regulator (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
142	D parameter for velocity PID (I)	D parameter of velocity PID regulator (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
143	I-Clipping parameter for velocity PID (I)	I-Clipping parameter of velocity PID (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
146	Activate ramp	1: Activate velocity ramp generator for position PID control. Allows usage of acceleration and positioning velocity for MVP command.	0/1	RWE
150	Actual motor current	Get actual motor current.	-2147483648… +2147483647 [mA]	R
151	Actual voltage	Actual supply voltage.	0… +4294967295	R
152	Actual driver temperature	Actual temperature of the motor driver.	0… +4294967295	R
153	Actual PWM duty cycle	Get actual PWM duty cycle.	-3599… +3599	R
154	Target PWM	Get desired target PWM or set target PWM to activate PWM regulation mode. (+= turn motor in right direction; -= turn motor in left direction)	-3599… +3599	RW
155	Target current	Get desired target current or set target current to activate current regulation mode. (+= turn motor in right direction; -= turn motor in left direction)	-2147483648… +2147483647 [mA]	RW

Number	Axis Parameter	Description	Range [Unit]	Access
156	Error/Status flags	Bit 0: Overcurrent flag. This flag is set if overcurrent limit is exceeded. Bit 1: Undervoltage flag. This flag is set if supply voltage to low for motor operation. Bit 2: Overvoltage flag. This flag is set if the motor becomes switched off due to overvoltage. Bit 3: Overtemperature flag. This flag is set if overtemperature limit is exceeded. Bit 4: Motor halted flag. This flag is set if motor has been switched off. Bit 5: Hall error flag. This flag is set upon a hall error. Bit 6: unused Bit 7: unused Bit 8: PWM mode active flag. Bit 9: Velocity mode active flag Bit 10: Position mode active flag. Bit 11: Torque mode active flag. Bit 12: unused Bit 13: unused Bit 14: Position end flag. This flag is set if the motor has been stopped at the target position. Bit 15: unused Bit 16: unused for TMCM-1640 (value should be 0 or 1) Bit 17: I^2t exceeded flag. This flag is set if the I^2t sum exceeded the I^2t limit of the motor. (reset by SAP 29 after the time specified by the I^2t thermal winding time constant) *Flag 0 to 15 are automatically reset. Only flag 16 and 17 must be cleared manually.*	0…+4294967295	R
159	Commutation mode	0: Block commutation with hall sensors 1: Sensorless block commutation (hallFX™) 3: Sine commutation with encoder	0, 1, 3	RWE
160	Re-Initialization of Sine	0: sine commutation is still re-initializing 1: sine commutation is re-initialized Attention: Depending on initialization mode, stop motor before issuing this command!	0/1	RW
161	Encoder set NULL	1: set position counter to zero at next N channel event.	0/1	RWE
162	Switch set NULL	1: set position counter to zero at next switch event.	0/1	RWE
163	Encoder clear set NULL	1: set position counter to zero only once 0: always at an N channel event, respectively switch event.	0/1	RWE

Number	Axis Parameter	Description	Range [Unit]	Access
164	Activate stop switch	Bit 0 – Left stop switch enable: When this bit is set the motor will be stopped if it is moving in negative direction and the left stop switch input becomes active Bit 1 – Right stop switch enable: When this bit is set the motor will be stopped if it is moving in positive direction and the right stop switch input becomes active Please see parameter 166 for selecting the stop switch input polarity.	0… 3	RWE
165	Actual encoder commutation offset	This value represents the internal commutation offset. (0 … max. encoder steps per rotation)	-2147483648… +2147483647	RWE
166	Stop switch polarity	Bit 0 – Left stop switch polarity: Bit set: Left stop switch input is high active Bit clear: Left stop switch input is low active Bit 1 – Right stop switch polarity: Bit set: Right stop switch input is high active Bit clear: Right stop switch input is low active	0… 3	RWE
167	Block PWM scheme	0: PWM chopper on high side, HI on low side 1: PWM chopper on low side, HI on high 2: PWM chopper on low side and high side	-128… +127	RWE
168	P parameter for current PID (I)	P parameter of current PID regulator. (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
169	I parameter for current PID (I)	I parameter of current PID regulator. (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
170	D parameter for current PID (I)	D parameter of current PID regulator. (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
171	I-Clipping parameter for current PID (I)	I-Clipping parameter of current PID regulator. (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
172	P parameter for current PID (II)	P parameter of current PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
173	I parameter for current PID (II)	I parameter of current PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
174	D parameter for current PID (II)	D parameter of current PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
175	I-Clipping parameter for current PID (II)	I-Clipping parameter of current PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
176	Threshold speed for current PID	Threshold speed for current regulation to switch between first and second current PID parameter set.	-2147483648… +2147483647 [rpm]	RWE
177	Start current	Motor current for controlled commutation. This parameter is used in commutation mode 1, 4, 5 and in initialization of sine.	0… +4294967295 [mA]	RWE

Number	Axis Parameter	Description	Range [Unit]	Access
200	Current PID error	Actual error of current PID regulator	-2147483648… +2147483647	R
201	Current PID error sum	Sum of errors of current PID regulator	-2147483648… +2147483647	R
209	Actual encoder position	Actual encoder position / counter value	-2147483648… +2147483647	R
226	Position PID error	Actual error of position PID regulator	-2147483648… +2147483647	R
227	Position PID error sum	Sum of errors of position PID regulator	-2147483648… +2147483647	R
228	Velocity PID error	Actual error of velocity PID regulator	-2147483648… +2147483647	R
229	Velocity PID error sum	Sum of errors of velocity PID regulator	-2147483648… +2147483647	R
230	P parameter for position PID (II)	P parameter of position PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
231	I parameter for position PID (II)	I parameter of position PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
232	D parameter for position PID (II)	D parameter of position PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
233	I-Clipping parameter for position PID (II)	I-Clipping parameter of position PID regulator. (second parameter set) (A too high value causes overshooting at positioning mode.)	-2147483648… +2147483647	RWE
234	P parameter for velocity PID (II)	P parameter of velocity PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
235	I parameter for velocity PID (II)	I parameter of velocity PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
236	D parameter for velocity PID (II)	D parameter of velocity PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
237	I-Clipping parameter for velocity PID (II)	I-Clipping parameter of velocity PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
238	Mass inertia constant	Mass inertia constant. Compensates the rotor inertia of the motor.	-2147483648… +2147483647	RWE
239	BEMF constant	BEMF constant of the motor. Used for current, position, and velocity regulation. Feed forward control for current, position, and velocity regulation is disabled if BEMF constant is set to zero.	-2147483648… +2147483647 [rpm/(10V)]	RWE
240	Motor coil resistance	Resistance of motor coil. Used for current, position, and velocity regulation.	-2147483648… +2147483647 [mΩ]	RWE
241	Init sine speed	Velocity for sine initialization. A positive sign initializes in right direction, a negative sign in left motor direction.	-32768… +32767 [rpm]	RWE
242	Init sine block offset CW	This parameter helps to tune hall sensor based initialization in a way, that the motor has the same velocity for left and right turn. It compensates for tolerance and hysteresis of the hall sensors. It is added to the Commutation offset upon CW turn initialization.	-32768… +32767	RWE

Number	Axis Parameter	Description	Range [Unit]	Access
243	Init sine block offset CCW	This parameter helps to tune hall sensor based initialization in a way, that the motor has the same velocity for left and right turn. It compensates for tolerance and hysteresis of the hall sensors. It is added to the Commutation offset upon CCW turn initialization.	-32768… +32767	RWE
244	Init sine delay	Duration for sine initialization sequence. This parameter should be set in a way, that the motor has stopped mechanical oscillations after the specified time.	-32768… +32767 [ms]	RWE
245	Overvoltage protection	1: Enable overvoltage protection.	0/1	RWE
247	Sine Compensation Factor	Compensates the propagation delay of the MPU	0… +255	RWE
249	Init sine mode	0: Initialization in controlled sine commutation (determines the encoder offset) 1: Initialization in block commutation using hall sensors 2: Initialization in controlled sine commutation (use the previous set encoder offset)	-128… +127	RWE
250	Encoder steps	Encoder steps per rotation.	0… +4294967295	RWE
251	Encoder direction	Set the encoder direction in a way, that ROR increases position counter.	0/1	RWE
253	Number of motor poles	Number of motor poles.	+2… +254	RWE
254	Hall sensor invert	1: Hall sensor invert. Invert the hall scheme, e.g. used by some Maxon motors.	0/1	RWE

6.1 Axis parameter sorted by functionality

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

Meaning of the letters in column Access:

R = readable (GAP)
W = writable (SAP)
E = stored and automatically restored from EEPROM after reset or power-on

MOTOR/MODULE SETTINGS

Number	Axis Parameter	Description	Range [Unit]	Access
253	Number of motor poles	Number of motor poles.	+2… +254	RWE
5	PWM limit	Set/get PWM limit (0%… 100%).	0… 3599	RWE
239	BEMF constant	BEMF constant of the motor. Used for current, position, and velocity regulation. Feed forward control for current, position, and velocity regulation is disabled if BEMF constant is set to zero.	-2147483648… +2147483647 [rpm/(10V)]	RWE
240	Motor coil resistance	Resistance of motor coil. Used for current, position, and velocity regulation.	-2147483648… +2147483647 [mΩ]	RWE
238	Mass inertia constant	Mass inertia constant. Compensates the rotor inertia of the motor.	-2147483648… +2147483647	RWE
136	PWM-Hysteresis	Compensates dead time of PWM and motor friction.	0… +4294967295	RWE
25	Thermal winding time constant	Thermal winding time constant for the used motor. Used for I^2t monitoring.	0… +4294967295 [ms]	RWE
26	I^2t limit	An actual I^2t sum that exceeds this limit leads to increasing the I^2t exceed counter.	0… +4294967295	RWE
27	I^2t sum	Actual sum of the I^2t monitor.	0… +4294967295	R
28	I^2t exceed counter	Counts how often an I^2t sum was higher than the I^2t limit.	0… +4294967295	RWE
29	Clear I^2t exceeded flag	Clear the flag that indicates that the I^2t sum has exceeded the I^2t limit.	(ignored)	W
30	Minute counter	Counts the module operational time in minutes.	0… +4294967295 [min]	RWE
245	Overvoltage protection	1: Enable overvoltage protection.	0/1	RWE

ENCODER/INITIALIZATION SETTINGS

Number	Axis Parameter	Description	Range [Unit]	Access
254	Hall sensor invert	1: Hall sensor invert. Invert the hall scheme, e.g. used by some Maxon motors.	0/1	RWE
250	Encoder steps	Encoder steps per rotation.	0… +4294967295	RWE
209	Actual encoder position	Actual encoder position / counter value	-2147483648… +2147483647	R
251	Encoder direction	Set the encoder direction in a way, that ROR increases position counter.	0/1	RWE
165	Actual encoder commutation offset	This value represents the internal commutation offset. (0 … max. encoder steps per rotation)	-2147483648… +2147483647	RWE
177	Start current	Motor current for controlled commutation. This parameter is used in commutation mode 1, 4, 5 and in initialization of sine.	0… +4294967295 [mA]	RWE

Number	Axis Parameter	Description	Range [Unit]	Access
249	Init sine mode	0: Initialization in controlled sine commutation (determines the encoder offset) 1: Initialization in block commutation using hall sensors 2: Initialization in controlled sine commutation (use the previous set encoder offset)	-128… +127	RWE
241	Init sine speed	Velocity for sine initialization. A positive sign initializes in right direction, a negative sign in left motor direction.	-32768… +32767 [rpm]	RWE
244	Init sine delay	Duration for sine initialization sequence. This parameter should be set in a way, that the motor has stopped mechanical oscillations after the specified time.	-32768… +32767 [ms]	RWE
14	velocity threshold for hallFX™	Velocity to switch from controlled to hallFX™ mode. Set this value to a realistic velocity which the motor can reach in controlled mode!	-2147483648… +2147483647 [rpm]	RWE
159	Commutation mode	0: Block commutation with hall sensors 1: Sensorless block commutation (hallFX™) 3: Sine commutation with encoder	0, 1, 3	RWE
160	Re-Initialization of Sine	0: sine commutation is still re-initializing 1: sine commutation is re-initialized Attention: Depending on initialization mode, stop motor before issuing this command!	0/1	RW
247	Sine Compensation Factor	Compensates the propagation delay of the MPU	0… +255	RWE
167	Block PWM scheme	0: PWM chopper on high side, HI on low side 1: PWM chopper on low side, HI on high 2: PWM chopper on low side and high side	-128… +127	RWE
242	Init sine block offset CW	This parameter helps to tune hall sensor based initialization in a way, that the motor has the same velocity for left and right turn. It compensates for tolerance and hysteresis of the hall sensors. It is added to the Commutation offset upon CW turn initialization.	-32768… +32767	RWE
243	Init sine block offset CCW	This parameter helps to tune hall sensor based initialization in a way, that the motor has the same velocity for left and right turn. It compensates for tolerance and hysteresis of the hall sensors. It is added to the Commutation offset upon CCW turn initialization.	-32768… +32767	RWE

PWM MODE

Number	Axis Parameter	Description	Range [Unit]	Access
5	PWM limit	Set/get PWM limit (0%… 100%).	0… 3599	RWE
153	Actual PWM duty cycle	Get actual PWM duty cycle.	-3599… +3599	R
154	Target PWM	Get desired target PWM or set target PWM to activate PWM regulation mode. (+= turn motor in right direction; -= turn motor in left direction)	-3599… +3599	RW

TORQUE REGULATION MODE

Number	Axis Parameter	Description	Range [Unit]	Access
6	Max current	Set/get the max allowed motor current. This value can be temporarily exceeded marginal due to the operation of the current regulator.	0… +4294967295 [mA]	RWE
150	Actual motor current	Get actual motor current.	-2147483648… +2147483647 [mA]	R
155	Target current	Get desired target current or set target current to activate current regulation mode. (+= turn motor in right direction; -= turn motor in left direction)	-2147483648… +2147483647 [mA]	RW
134	Current regulation loop delay	Delay of current limitation algorithm / PID current regulator.	0… +4294967295 [50µs]	RWE
176	Threshold speed for current PID	Threshold speed for current regulation to switch between first and second current PID parameter set.	-2147483648… +2147483647 [rpm]	RWE
168	P parameter for current PID (I)	P parameter of current PID regulator. (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
169	I parameter for current PID (I)	I parameter of current PID regulator. (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
170	D parameter for current PID (I)	D parameter of current PID regulator. (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
171	I-Clipping parameter for current PID (I)	I-Clipping parameter of current PID regulator. (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
172	P parameter for current PID (II)	P parameter of current PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
173	I parameter for current PID (II)	I parameter of current PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
174	D parameter for current PID (II)	D parameter of current PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
175	I-Clipping parameter for current PID (II)	I-Clipping parameter of current PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
200	Current PID error	Actual error of current PID regulator	-2147483648… +2147483647	R
201	Current PID error sum	Sum of errors of current PID regulator	-2147483648… +2147483647	R

Velocity regulation mode

Number	Axis Parameter	Description	Range [Unit]	Access
3	Actual speed	The actual velocity of the motor.	-2147483648… +2147483647 [rpm]	R
2	Target speed	Set/get the desired target velocity.	-2147483648… +2147483647 [rpm]	RW
9	Motor halted velocity	If the actual speed is below this value the motor halted flag will be set.	-2147483648… +2147483647 [rpm]	RWE
133	PID regulation loop delay	PID calculation delay. Set PID operational frequency.	0… +4294967295 [ms]	RWE
8	Threshold speed for velocity PID	Threshold speed for velocity regulation to switch between first and second velocity PID parameter set.	-2147483648… +2147483647 [rpm]	RWE
140	P parameter for velocity PID (I)	P parameter of velocity PID regulator (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
141	I parameter for velocity PID (I)	I parameter of velocity PID regulator (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
142	D parameter for velocity PID (I)	D parameter of velocity PID regulator (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
143	I-Clipping parameter for velocity PID (I)	I-Clipping parameter of velocity PID (first parameter set, used at lower speed)	-2147483648… +2147483647	RWE
234	P parameter for velocity PID (II)	P parameter of velocity PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
235	I parameter for velocity PID (II)	I parameter of velocity PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
236	D parameter for velocity PID (II)	D parameter of velocity PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
237	I-Clipping parameter for velocity PID (II)	I-Clipping parameter of velocity PID regulator. (second parameter set, used at higher speed)	-2147483648… +2147483647	RWE
228	Velocity PID error	Actual error of PID velocity regulator	-2147483648… +2147483647	R
229	Velocity PID error sum	Sum of errors of PID velocity regulator	-2147483648… +2147483647	R

Velocity ramp parameter

Number	Axis Parameter	Description	Range [Unit]	Access
4	Max. ramp velocity	The maximum velocity used for velocity ramp in velocity mode and positioning mode. Set this value to a realistic velocity which the motor can reach!	-2147483648… +2147483647 [rpm]	RWE
11	Acceleration	Acceleration parameter for ROL, ROR, and the velocity ramp of MVP.	-2147483648… +2147483647 [RPM/s]	RWE
13	Ramp generator speed	The actual speed of the velocity ramp used for positioning and velocity mode.	-2147483648… +2147483647 [rpm]	R
146	Activate ramp	1: Activate velocity ramp generator for position PID control. Allows usage of acceleration and positioning velocity for MVP command.	0/1	RWE

Position regulation mode

Number	Axis Parameter	Description	Range [Unit]	Access
1	Actual position	Set/get the position counter without moving the motor.	-2147483648… +2147483647	RW
0	Target position	The target position of a currently executed ramp.	-2147483648… +2147483647	RW
7	MVP Target reached velocity	Maximum velocity at which end position can be set. Prevents issuing of end position when the target is passed at high velocity.	-2147483648… +2147483647 [rpm]	RWE
10	MVP target reached distance	Maximum distance at which the position end flag is set.	-2147483648… +2147483647	RWE
161	Encoder set NULL	1: set position counter to zero at next N channel event.	0/1	RWE
162	Switch set NULL	1: set position counter to zero at next switch event.	0/1	RWE
163	Encoder clear set NULL	1: set position counter to zero only once 0: always at an N channel event, respectively switch event.	0/1	RWEP
12	Threshold speed for position PID	Threshold speed for position regulation to switch between first and second position PID parameter set.	-2147483648… +2147483647 [rpm]	RWE
130	P parameter for position PID (I)	P parameter of position PID regulator (first parameter set)	-2147483648… +2147483647	RWE
131	I parameter for position PID (I)	I parameter of position PID regulator (first parameter set)	-2147483648… +2147483647	RWE
132	D parameter for position PID (I)	D parameter of position PID regulator (first parameter set)	-2147483648… +2147483647	RWE
135	I-Clipping parameter for position PID (I)	I-Clipping parameter of position PID regulator (first parameter set) (A too high value causes overshooting at positioning mode.)	-2147483648… +2147483647	RWE
230	P parameter for position PID (II)	P parameter of position PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
231	I parameter for position PID (II)	I parameter of position PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
232	D parameter for position PID (II)	D parameter of position PID regulator. (second parameter set)	-2147483648… +2147483647	RWE
233	I-Clipping parameter for position PID (II)	I-Clipping parameter of position PID regulator. (second parameter set) (A too high value causes overshooting at positioning mode.)	-2147483648… +2147483647	RWE
226	Position PID error	Actual error of PID position regulator	-2147483648… +2147483647	R
227	Position PID error sum	Sum of errors of PID position regulator	-2147483648… +2147483647	R

STATUS INFORMATION

Number	Axis Parameter	Description	Range [Unit]	Access
151	Actual voltage	Actual supply voltage.	0… +4294967295	R
152	Actual driver temperature	Actual temperature of the motor driver.	0… +4294967295	R
156	Error/Status flags	Bit 0: Overcurrent flag. This flag is set if overcurrent limit is exceeded. Bit 1: Undervoltage flag. This flag is set if supply voltage to low for motor operation. Bit 2: Overvoltage flag. This flag is set if the motor becomes switched off due to overvoltage. Bit 3: Overtemperature flag. This flag is set if overtemperature limit is exceeded. Bit 4: Motor halted flag. This flag is set if motor has been switched off. Bit 5: Hall error flag. This flag is set upon a hall error. Bit 6: unused Bit 7: unused Bit 8: PWM mode active flag. Bit 9: Velocity mode active flag Bit 10: Position mode active flag. Bit 11: Torque mode active flag. Bit 12: unused Bit 13: unused Bit 14: Position end flag. This flag is set if the motor has been stopped at the target position. Bit 15: unused Bit 16: unused for TMCM-1640 (value should be 0 or 1) Bit 17: I^2t exceeded flag. This flag is set if the I^2t sum exceeded the I^2t limit of the motor. (reset by SAP 29 or after the time specified by the I^2t thermal winding time constant) *Flag 0 to 15 are automatically reset. Only flag 16 and 17 must be cleared manually.*	0…+4294967295	R
157	Module supply current	Get actual supply current of the module.	0… +4294967295 [mA]	R

SWITCHES AND ANALOG INPUTS

Number	Axis Parameter	Description	Range [Unit]	Access
20	Switch 2 active	0: inactive 1: active (IO connector pin 4)	0/1	R
21	Switch 1 active	0: inactive 1: active (IO connector pin 5)	0/1	R
164	Activate stop switch	Bit 0 – Left stop switch enable – When this bit is set the motor will be stopped if it is moving in negative direction and the left stop switch input becomes active Bit 1 – Right stop switch enable – When this bit is set the motor will be stopped if it is moving in positive direction and the right stop switch input becomes active Please see parameter 166 for selecting the stop switch input polarity.	0… 3	RWE
166	Stop switch polarity	Bit 0 – Left stop switch polarity – Bit set: Left stop switch input is high active Bit clear: Left stop switch input is low active Bit 1 – Right stop switch polarity – Bit set: Right stop switch input is high active Bit clear: Right stop switch input is low active	0… 3	RWE

7 Global parameter overview (SGP, GGP, STGP, RSGP)

The following section describes all global parameters that can be used with the SGP, GGP, STGP and RSGP commands.

Global parameters are grouped into 3 banks:

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

7.1 Bank 0

Parameters 64… 255
Parameters below 63 configure stuff like the serial address of the module RS485 baud rate or the telegram pause time. 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 between 64 and 85 are stored in EEPROM only. A SGP command on such a parameter will always store it permanently and no extra STGP command is needed.

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

Meaning of the letters in column *Access*:

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

<table>
<tr><th>Number</th><th>Global parameter</th><th>Description</th><th>Range</th><th>Access</th></tr>
<tr><td>64</td><td>EEPROM magic</td><td>Setting this parameter to a different value as $E4 will cause re-initialization of the axis and global parameters (to factory defaults) after the next power up. This is useful in case of miss-configuration.</td><td>0… 255</td><td>RWE</td></tr>
<tr><td>65</td><td>RS485 baud rate</td><td><table>
<tr><td>0</td><td>9600 baud</td><td>Default</td></tr>
<tr><td>1</td><td>14400 baud</td><td></td></tr>
<tr><td>2</td><td>19200 baud</td><td></td></tr>
<tr><td>3</td><td>28800 baud</td><td></td></tr>
<tr><td>4</td><td>38400 baud</td><td></td></tr>
<tr><td>5</td><td>57600 baud</td><td></td></tr>
<tr><td>6</td><td>76800 baud</td><td>Not supported by Windows!</td></tr>
<tr><td>7</td><td>(115200 baud)</td><td></td></tr>
</table></td><td>0… 7</td><td>RWE</td></tr>
<tr><td>66</td><td>serial address</td><td>The module (target) address for RS485 and virtual COM port</td><td>0… 255</td><td>RWE</td></tr>
<tr><td>73</td><td>configuration EEPROM lock flag</td><td>Write: 1234 to lock the EEPROM, 4321 to unlock it.
Read: 1=EEPROM locked, 0=EEPROM unlocked.</td><td>0/1</td><td>RWE</td></tr>
<tr><td>75</td><td>telegram pause time</td><td>Pause time before the reply via RS485 is sent.</td><td>0… 255</td><td>RWE</td></tr>
<tr><td>76</td><td>serial host address</td><td>Host address used in the reply telegrams sent back via RS485.</td><td>0… 255</td><td>RWE</td></tr>
<tr><td>77</td><td>auto start mode</td><td>0: Do not start TMCL™ application after power up (default).
1: Start TMCL™ application automatically after power up.</td><td>0/1</td><td>RWE</td></tr>
</table>

Number	Global parameter	Description	Range	Access
81	TMCL™ code protection	Protect a TMCL™ program against disassembling or overwriting. 0 – no protection 1 – protection against disassembling 2 – protection against overwriting 3 – protection against disassembling and overwriting *If you switch off the protection against disassembling, the program will be erased first!* *Changing this value from 1 or 3 to 0 or 2, the TMCL™ program will be wiped off.*	0, 1, 2, 3	RWE
85	do not restore user variables	0 – user variables are restored (*default*) 1 – user variables are not restored	0/1	RWE
128	TMCL™ application status	0 –stop 1 – run 2 – step 3 – reset	0… 3	R
129	download mode	0 – normal mode 1 – download mode	0/1	R
130	TMCL™ program counter	The index of the currently executed TMCL™ instruction.	o… 2047	R
132	tick timer	A 32 bit counter that gets incremented by one every millisecond. It can also be reset to any start value.	0… +4294967295	RW
255	suppress reply	0 – reply (*default*) 1 – no reply	0/1	RW

7.2 Bank 1

The global parameter bank 1 is normally not available. It may be used for customer specific extensions of the firmware. Together with user definable commands (see section 0) these variables form the interface between extensions of the firmware (written in C) and TMCL™ applications.

7.3 Bank 2

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

Up to 256 user variables are available.

Meaning of the letters in column *Access*:

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

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

8 PID regulation

8.1 Structure of the cascaded motor regulation modes

The TMC603-EVAL supports current, velocity, and position PID regulation modes for motor control in different application areas. These regulation modes are cascaded as shown in Figure 8.1. The specific modes are explained in the following sections.

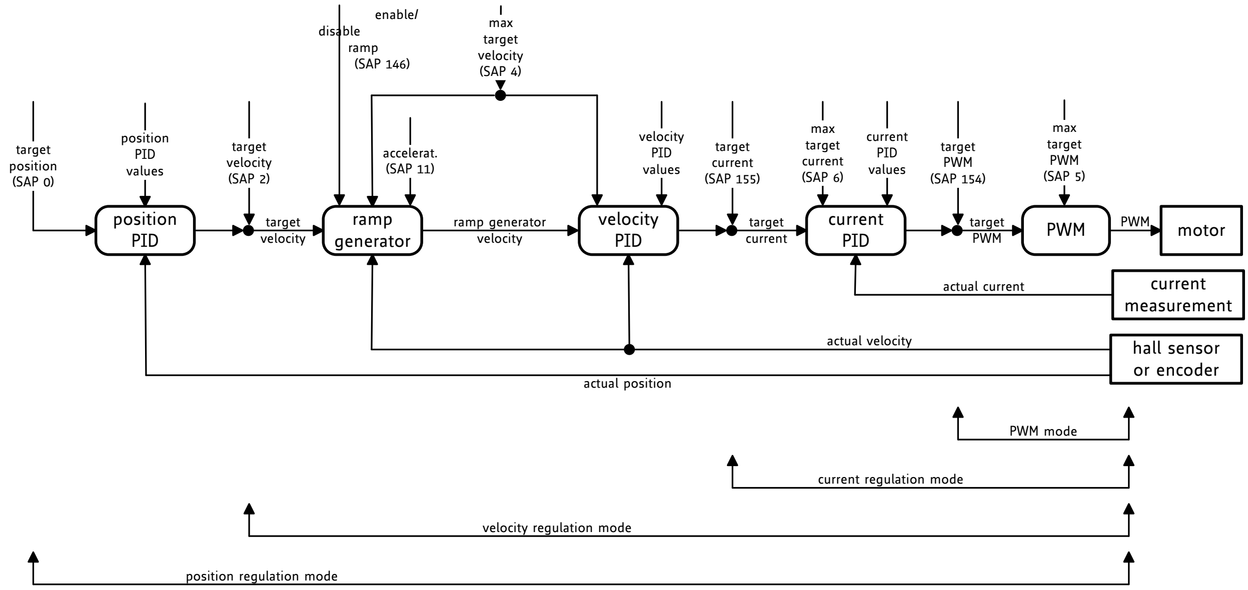


Figure 8.1: Cascaded PID regulation

8.2 PWM regulation

The PWM regulation mode is the most direct control mode for the TMCM-1632/TMCM-KR-841. Thereby, a target PWM given by axis parameter 154 is adjusted directly **without limiting the motor current**. The target PWM is only limited by axis parameter 5 (max target PWM). The sign of the target PWM controls the motor rotation direction.

8.3 Current PID regulation

Based on the PWM regulation the current regulation mode uses a PID regulator to adjust a desired motor current. This target current can be set by axis parameter 155. The maximal current is limited by axis parameter 6.

The PID regulation uses five basic parameters: The *P*, *I*, *D* and *I-Clipping value* as well as the *timing control value*. The timing control value (*current regulation loop delay*, axis parameter 134) determines how often the current regulation is invoked. It is given in multiple of 50µs:

$$t_{PIDDELAY} = x_{PIDRLD} \cdot 50\mu s$$

$t_{PIDDELAY}$ = resulting delay between two PID calculations
x_{PIDRLD} = *current regulation loop delay* parameter

For most applications it is recommended to leave this parameter unchanged at its default of 50µs. Higher values may be necessary for very slow and less dynamic drives. The structure of the current PID regulator is shown in Figure 8.2. It has to be parameterized with respect to a given motor.

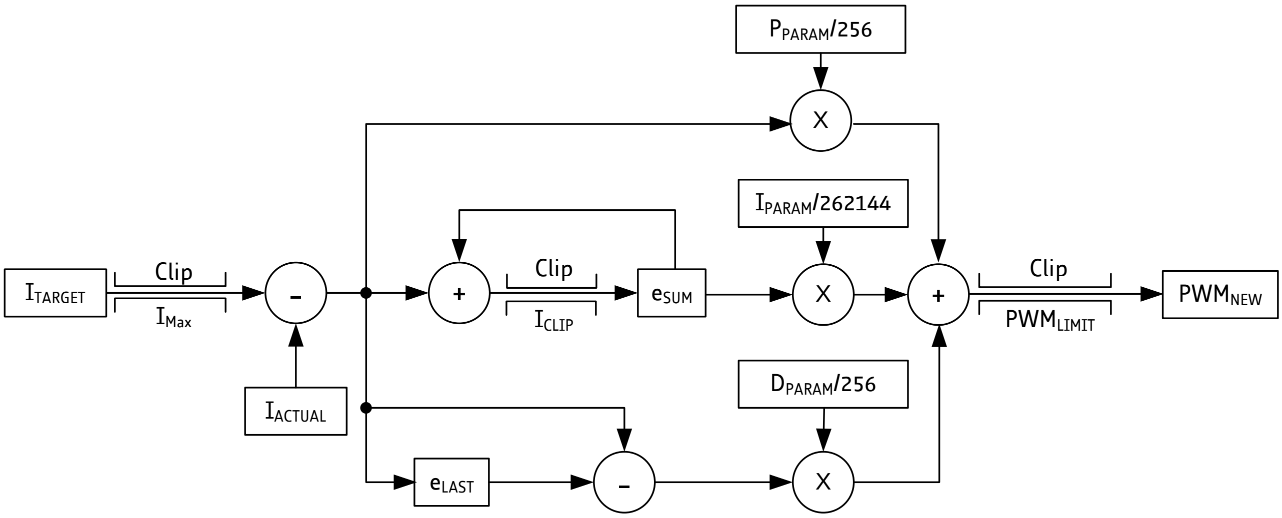


Figure 8.2: Current PID regulation

Parameter	Description
I_{ACTUAL}	Actual motor current (GAP 150)
I_{TARGET}	Target motor current (SAP 155)
I_{Max}	Max. motor current (SAP 6)
e_{LAST}	Error value of the last PID calculation (GAP 200)
e_{SUM}	Error sum for integral calculation (GAP 201)
P_{PARAM}	Current P parameter (SAP 168, SAP 172)
I_{PARAM}	Current I parameter (SAP169, SAP 173)
D_{PARAM}	Current D parameter (SAP 170, SAP 174)
I_{CLIP}	Current I-Clipping parameter (SAP 171, SAP175)
PWM_{LIMIT}	PWM Limit (SAP 5)
PWM_{NEW}	New target PWM value (GAP 153)

To parameterize the current PID regulator for a given motor, first set the P, I and D parameter of both parameter sets to zero. Then start the motor by using a low target current (e.g. 1000mA). Then modify the *current P parameter (II)*. This is the P parameter of parameter set 2. Start from a low value and go to a higher value, until the actual current nearly reaches the desired target current.
After that, do the same for the *current I parameter (II)* with the *current D parameter (II)* still set to zero. For the *current I parameter (II)*, there is also a clipping value. The *current I clipping parameter (II)* should be set to a relatively low value to avoid overshooting, but high enough to reach the target current. The *current D parameter (II)* can still be set to zero.
After having found suitable values for parameter set 2, the first parameter set (PID Parameter Set 1) should be set to lower values, to minimize overshooting during zero-time of motor start. Then stop the motor and start again to test the current regulation settings. If the motor current is overshot during zero-time, set the PID parameter set 1 once more to lower values.

For all tests set the motor current limitation to a realistic value, so that your power supply does not become overloaded during acceleration phases. If your power supply goes to current limitation, the unit may reset or undetermined regulation results may occur.

8.4 Velocity PID regulation

Based on the current regulation the motor velocity can be controlled by the velocity PID regulator. Also, the velocity PID regulator uses a timing control value (*PID regulation loop delay*, axis parameter 133) which determines how often the PID regulator is invoked. It is given in multiple of 1ms:

$$t_{PIDDELAY} = x_{PIDRLD} \cdot 1\text{ms}$$

$t_{PIDDELAY}$ = resulting delay between two PID calculations
x_{PIDRLD} = *PID regulation loop delay* parameter

For most applications it is recommended to leave this parameter unchanged at its default of 1ms. Higher values may be necessary for very slow and less dynamic drives. The structure of the velocity PID regulator is shown in Figure 8.3.

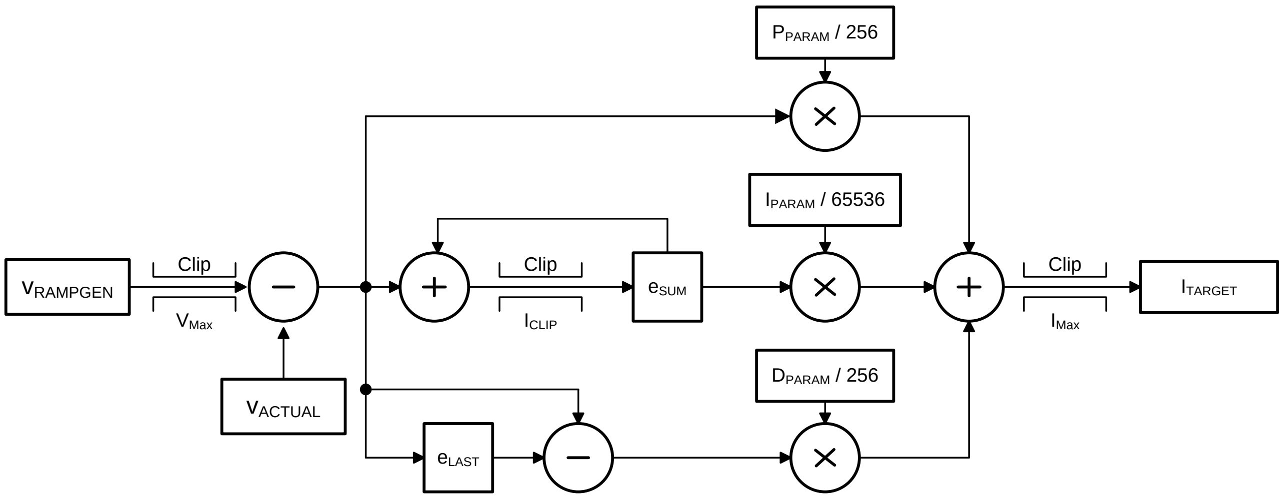


Figure 8.3: Velocity PID regulation

Parameter	Description
v_{ACTUAL}	Actual motor velocity (GAP 3)
$v_{RAMPGEN}$	Target velocity of ramp generator (SAP 2, GAP 13)
v_{Max}	Max. target velocity (SAP 4)
e_{LAST}	Error value of the last PID calculation (GAP 228)
e_{SUM}	Error sum for integral calculation (GAP 229)
P_{PARAM}	Velocity P parameter (SAP 140, SAP 234)
I_{PARAM}	Velocity I parameter (SAP 141, SAP 235)
D_{PARAM}	Velocity D parameter (SAP 142, SAP 236)
I_{CLIP}	Velocity I-Clipping parameter (SAP 143, SAP 237)
I_{Max}	Max. target current (SAP 6)
I_{Target}	Target current for current PID regulator (GAP 155)

For parameterizing the PID regulator set the *velocity I parameter* and *velocity D parameter* to zero and start the motor by using a medium target velocity (e.g. 3000 rpm). Then modify the *velocity P parameter*, only. Start from a low value and go to a higher value, until the actual motor speed reaches 80 or 90% of the desired motor speed. The rest of the speed difference can be reduced by using a high I clipping value (e.g. 500000) and a slow increase of the *velocity I parameter* with the *velocity D parameter* still set to zero. For the first tests, both PID parameter sets can be set equal.

8.5 Velocity ramp generator

For a controlled start up of the motor's velocity a velocity ramp generator can be activated/deactivated by axis parameter 146. The ramp generator uses the maximal allowed motor velocity (axis parameter 4), the acceleration (axis parameter 11) und the desired target velocity (axis parameter 2) to calculate a ramp generator velocity for the following velocity PID regulator.

8.6 Position PID regulation

Based on the current and velocity PID regulators the TMC603-EVAL supports a positioning mode based on encoder or hall sensor position. During positioning the velocity ramp generator can be activated to enable motor positioning with controlled acceleration or disabled to support motor positioning with max allowed speed. The structure of the position PID regulator is shown in Figure 8.4.

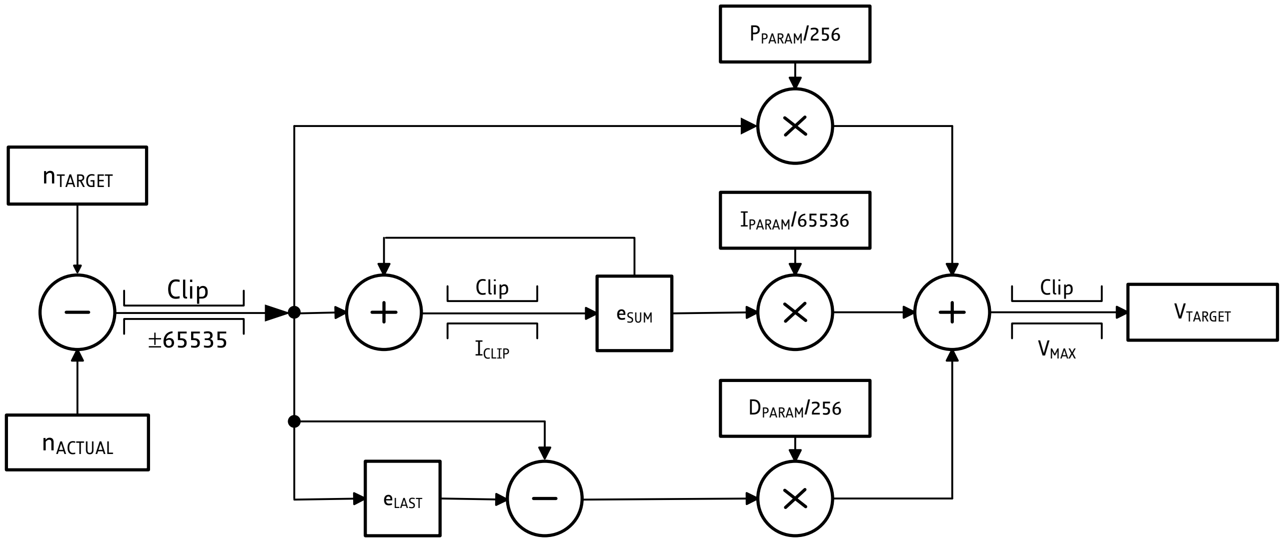


Figure 8.4: Positioning PID regulation

Parameter	Description
n_{ACTUAL}	Actual motor position (GAP 1)
n_{TARGET}	Target motor position (SAP 0)
e_{LAST}	Error value of the last PID calculation (GAP 226)
e_{SUM}	Error sum for integral calculation (GAP 227)
P_{PARAM}	Position P parameter (SAP 130, SAP 230)
I_{PARAM}	Position I parameter (SAP 131, SAP 231)
D_{PARAM}	Position D parameter (SAP 132, SAP 232)
I_{CLIP}	Position I-Clipping parameter (SAP 135, SAP 233)
V_{MAX}	Max. allowed velocity (SAP 4)
V_{TARGET}	New target velocity for ramp generator (GAP 13)

The PID regulation uses five basic parameters. The P, I, D, and I-Clipping value as well as a timing control value. The timing control value (*PID regulation loop delay* - axis parameter 133) determines how often the PID regulator is invoked. It is given in multiple of 1ms:

$$t_{PIDDELAY} = x_{PIDRLD} \cdot 1\text{ms}$$

$t_{PIDDELAY}$ = the resulting delay between two PID calculations

x_{PIDRLD} = *PID regulation loop delay* parameter

For most applications it is recommended to leave this parameter unchanged at its default of 1ms. Higher values may be necessary for very slow and less dynamic drives.

Based on the velocity PID regulator the position PID regulator can be parameterized as P regulator in the simplest case. Therefore, disable the velocity ramp generator and set *position P, I, and D parameters* to zero. Now, set a target position and increase the *position P parameter* until the motor reaches the target position approximately.

After finding a good *position P parameter* the velocity ramp generator can be switched on again. Based on the *max. positioning velocity* (axis parameter 4) as well as the *acceleration* (axis parameter 11) value the ramp generator automatically calculates the slow down point, i.e. the point at which velocity is to be reduced in order to stop at the desired target position. Reaching the target position is signaled by setting the *position end flag*.

In order to minimize the time until this flag becomes set, a positioning tolerance (*MVP target reached distance*) can be chosen by axis parameter 10. Since the motor typically is assumed not to signal target reached when the target was just passed in a short moment at a high velocity, additionally a maximum target reached velocity (*MVP target reached velocity*) can be defined by axis parameter 7. A value of zero is the most universal, since it implies that the motor stands still at the target. But when a fast rising of the *Position end flag* is desired, a higher value for *MVP target reached velocity* parameter will save a lot of time. The best value should be tried out in the actual application. The correlation of axis parameter 10, 7, the target position and the position end flag are summarized in Figure 8.5.

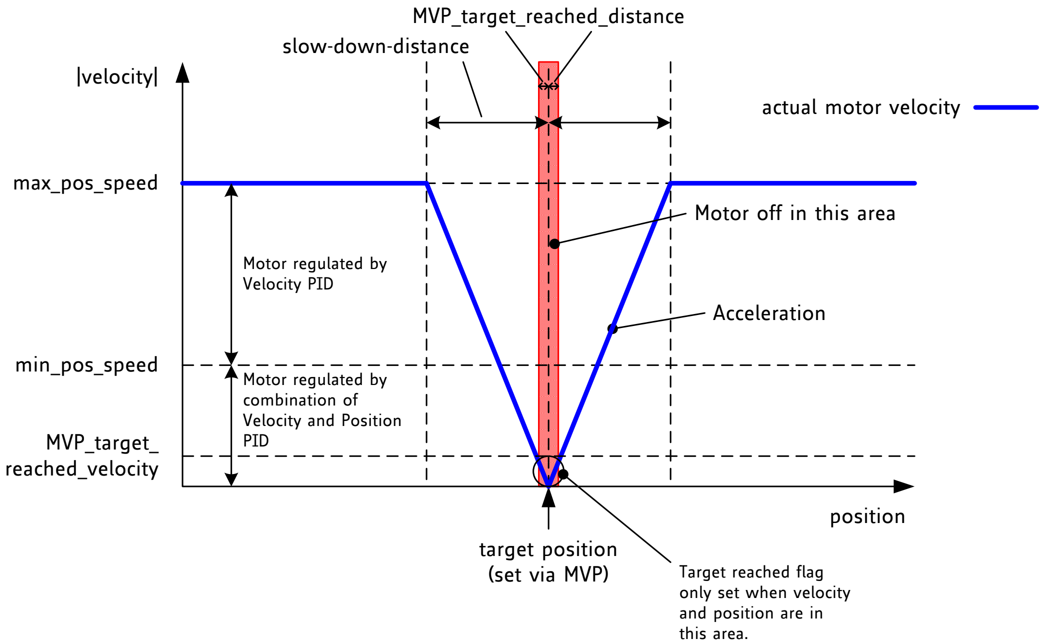


Figure 8.5: Positioning algorithm

Depending on the motor and mechanics respectively, a bit of oscillation is normal, in the best case it can be reduced to be at least +/-1 encoder step, because otherwise the regulation cannot keep the position.

8.7 Parameter sets for PID regulation

Every PID regulation provides two parameter sets, which are used as follows:

- Below a specified velocity threshold the PID regulator uses a combination of parameter set 1 and parameter set 2
- Above the velocity threshold the PID regulator uses only parameter set 2. If the velocity threshold is set to zero, parameter set 2 is used all the time. (The switch over between both parameter sets is soft.)

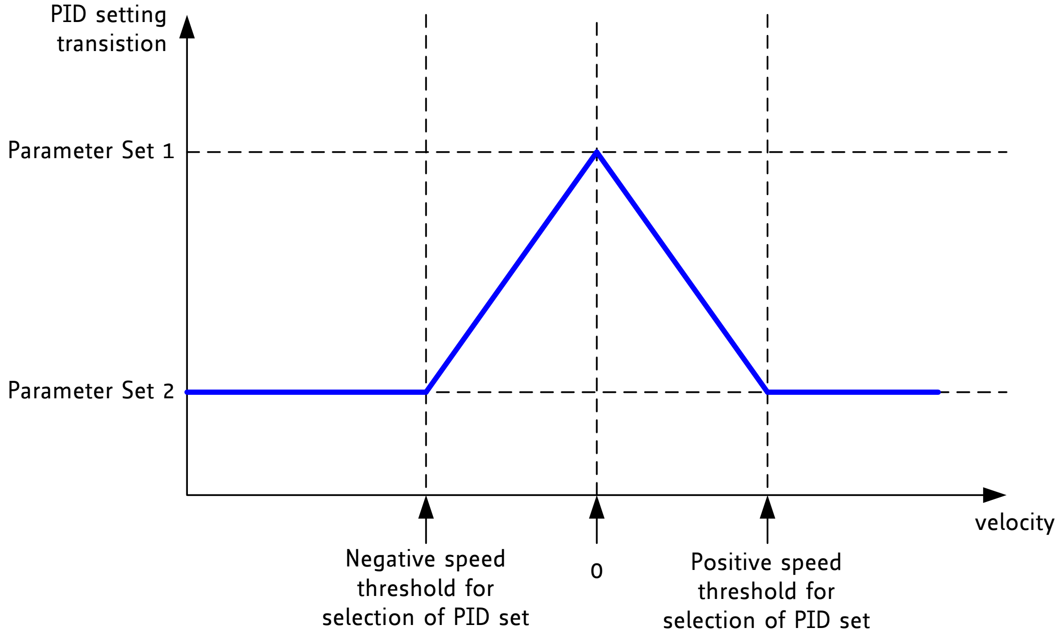


Figure 8.5: Transition between PID parameter sets

The velocity thresholds for current, velocity, and position PID regulation can be set as follows:

- axis parameter 176: velocity threshold for current PID regulator
- axis parameter 8: velocity threshold for velocity PID regulator
- axis parameter 12: velocity threshold for position PID regulator

Attention: For all tests set the motor current limitation to a realistic value, so that your power supply does not become overloaded during acceleration phases. If your power supply goes to current limitation, the unit may reset or undetermined regulation results may occur.

9 Temperature calculation

Axis parameter 152 delivers the actual ADC value of the motor driver. This ADC value can be converted to a temperature in °C as follows:

$$\mathrm{ADC} = \textit{actual value of GAP } 152$$

$$\mathrm{B} = 3434 \ (\textit{material constant})$$

$$\mathrm{R_{NTC}} = \frac{9011{,}2}{ADC} - 2.2$$

$$\mathrm{T} = \frac{B * 298{,}16}{B + (\ln\left(\frac{R_{NTC}}{10}\right) * 298.16} - 273.16\ ℃$$

Example 1:
ADC = 1000
R_{NTC} ≈ 6.81
T ≈ 35℃

Example 2:
ADC = 1200
R_{NTC} ≈ 5.31
T ≈ 42℃

10 I²t monitoring

The I²t monitor determines the sum of the square of the motor current over a given time. The integrating time is motor specific. In the datasheet of the motor this time is described as *thermal winding time constant* and can be set for each module using axis parameter 25. The number of measurement values within this time depends on how often the current regulation and thus the I²t monitoring is invoked. The value of the actual I²t sum can be read by axis parameter 27. With axis parameter 26 the default value for the I²t limit can be changed (default: 211200). If the actual I²t sum exceeds the I²t limit of the motor, flag 17 (in axis parameter 156) is set and the motor PWM is set to zero as long as the I²t exceed flag is set. The actual regulation mode will not be changed. Furthermore, the I²t exceed counter is increased once every second as long as the actual I²t sum exceeds the I²t limit. The I²t exceed flag can be cleared manually using parameter 29 but only after the cool down time given by the *thermal winding time constant* has passed. The I²t exceed flag will not be reset automatically. The I²t limit can be determined as follows:

$$I^2t = \frac{I\,[mA]}{1000} * \frac{I\,[mA]}{1000} * t_{tw}\,[ms]$$

I is the desired average current
t_{tw} is the thermal winding time constant given by the motor datasheet

Example:

I²t limits for an average current of a) 1A, b) 2A, c) 3A and d) 4A over a thermal winding time of 13,2s.

a) $I^2t\ limit = \frac{1000\,[mA]}{1000} * \frac{1000\,[mA]}{1000} * 13200\,[ms] = 13200\,[mA^2 * ms]$

b) $I^2t\ limit = \frac{2000\,[mA]}{1000} * \frac{2000\,[mA]}{1000} * 13200\,[ms] = 52800\,[mA^2 * ms]$

c) $I^2t\ limit = \frac{3000\,[mA]}{1000} * \frac{3000\,[mA]}{1000} * 13200\,[ms] = 118800\,[mA^2 * ms]$

d) $I^2t\ limit = \frac{4000\,[mA]}{1000} * \frac{4000\,[mA]}{1000} * 13200\,[ms] = 211200\,[mA^2 * ms]$

11 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.

12 Revision history

12.1 Firmware revision

Version	Date	Author OK – Olav Kahlbaum ED – Enrico Dressler	Description
1.29	2010-SEP-14	OK	First version
1.46	2011-SEP-27	ED	New version including hall*FX*™ parameters
1.47	2011-DEC-12	ED	New version with updated axis parameters

12.2 Document revision

Version	Date	Author SD – Sonja Dwersteg	Description
1.00	2011-FEB-14	SD	Initial version
1.01	2011-FEB-17	SD	Ranges of some axis parameters corrected
1.20	2011-NOV-07	SD	Complete revision including parameter descriptions for hallFX™.
1.21	2011-DEC-12	SD	- Axis parameters 5, 150, 153, 159, and 154 corrected - Axis parameter 246 deleted - Chapter 10 updated (I^2t monitoring) - Chapter 8 updated (PID regulation)
1.22	2011-JAN-02	SD	Minor changes
1.23	2012-AUG-29	SD	Axis parameters 164 and 166 added

13 References

[TMCM-1640] TMCM-1640 Hardware Manual
[TMCL-IDE] TMCL-IDE User Manual
[TMCL-BLDC] TMCL-BLDC User Manual
[TMC603] TMC603 Datasheet
[QBL4208] QBL4208 Manual

Please refer to our homepage http://www.trinamic.com.