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

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

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

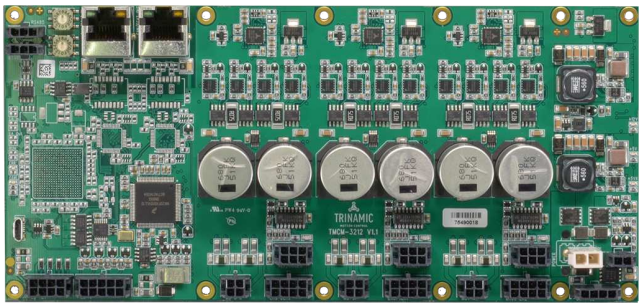
Contact us

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

TMCM-3213 CoE Firmware Manual

Firmware Version V1.01 | Document Revision V1.02 • 2016-NOV-29

The TMCM-3213 is a three axes controller/driver module for 2-phase bipolar stepper motors with separate differential encoder and separate home and stop switch inputs for each axis. Dynamic current control, and quiet, smooth and efficient operation are combined with stealthChop™, dcStep™, stallGuard™ and coolStep™ features. The module offers four analog or digital inputs as well as four digital outputs in combination with a break chopper unit.



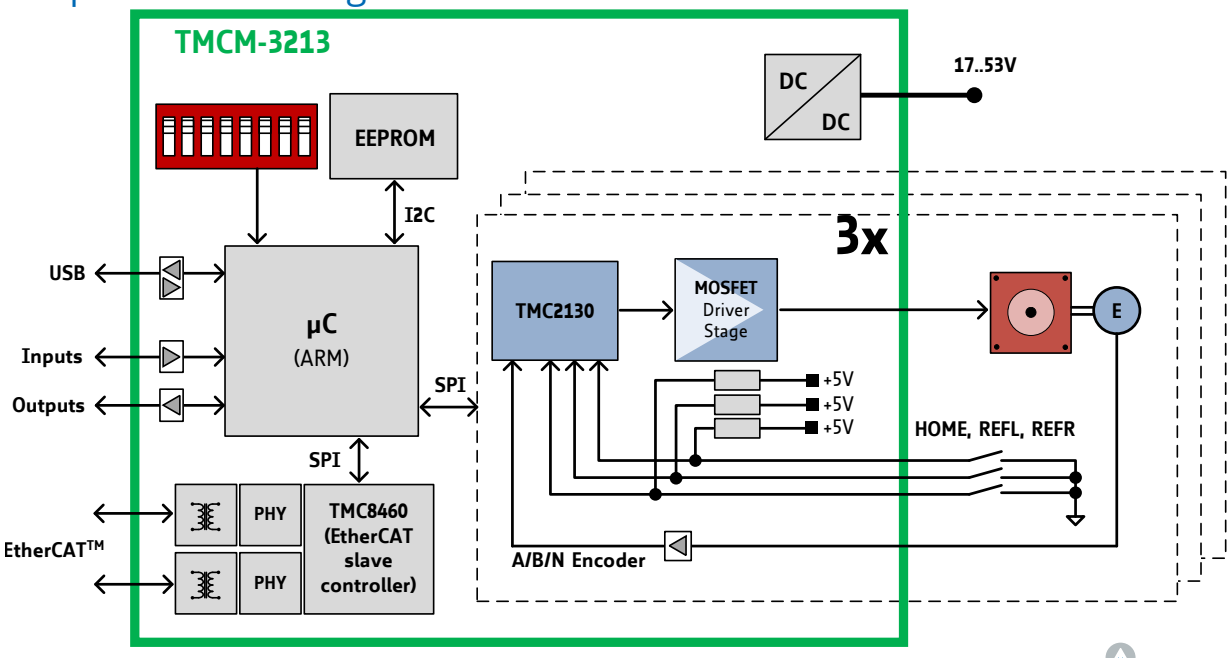
Features

- 3-Axes Stepper Motor Control
- CoE CiA-402 Drive Profile
- Encoder Support
- coolStep™
- dcStep™
- stallGuard2™
- stealthChop™

Applications

- Lab-Automation
- Semiconductor Handling
- Manufacturing
- Robotics
- Factory Automation
- CNC
- Laboratory Automation

Simplified Block Diagram



Read entire documentation.

TRINAMIC
MOTION CONTROL

Contents

1 Preface

This document specifies objects and modes of operation of the Triamic TMCM-3213 stepper motor control module with CANopen-over-EtherCAT (CoE) firmware. The CoE firmware is designed to fulfill the EtherCAT version of the CANopen DS402 standards. The EtherCAT conformance has also been tested. This manual assumes that the reader is already familiar with the basics of EtherCAT and the CoE protocol (especially DS402).
If necessary it is always possible to turn the module into a TMCL module by loading the TMCM-3213 TMCL fimware again through the USB interface, with the help of the firmware update function of the TMCL-IDE 3.0.

1.1 General Features of this CoE Implementation

Main Characteristics

- Communication according to EtherCAT standards
- Protocols: CoE, FoE

SDO Communication

- 1 server
- Expedited transfer
- Segmented transfer
- No block transfer

PDO Communication

- Producer
- Consumer
- RPDOs
  - Dynamic mapping with max. 9 mapping entries.
  - Default mappings: manufacturer specific.
- TPDOs
  - Dynamic mapping with max. 9 mapping entries.
  - Default mappings: manufacturer specific.

Sync managers

- Sync manager 0: receive mailbox used for SDO communication
- Sync manager 1: send mailbox used for SDO communication
- Sync manager 2: process data output (used for TPDO)
- Sync manager 3: process data input (used for RPDO)

Further Characteristics

- Emergency: producer

1.2 Abbreviations used in this Manual

Abbreviations	
CAN	Controller area network
CoE	CANopen over EtherCAT
CHGND	chassis ground / earth ground
COB	Communication object
FoE	File transfer over EtherCAT
FSA	Finite state automaton
FSM	Finite state machine
NMT	Network management
ID	Identifier
LSB	Least significant bit
MSB	Most significant bit
PDO	Process data object
PDS	Power drive system
RPDO	Receive process data object
SDO	Service data object
TPDO	Transmit process data object
EMCY	Emergency object
rw	Read and write
ro	Read only
hm	Homing mode
pp	Profile position mode
pv	Profile velocity mode
vm	Velocity mode

Table 1: Abbreviations used in this Manual

1.3 Firmware Update

The software running on the microprocessor consists of two parts, a boot loader and the CANopen firmware itself. Whereas the boot loader is installed during production and testing at TRINAMIC and remains untouched throughout the whole lifetime, the CANopen firmware can easily be updated by the user. The new firmware can be loaded into the module via the firmware update function of the TMCL-IDE, using the USB interface of the module.

1.4 Trinamic's unique Features — easy to use with CoE

1.4.1 stallGuard2

stallGuard2 is a high-precision sensorless load measurement using the back EMF of the coils. It can be used for stall detection as well as other uses at loads below those which stall the motor. The stallGuard2 measurement value changes linearly over a wide range of load, velocity, and current settings. At maximum motor load, the value reaches zero or is near zero. This is the most energy-efficient point of operation for the motor.

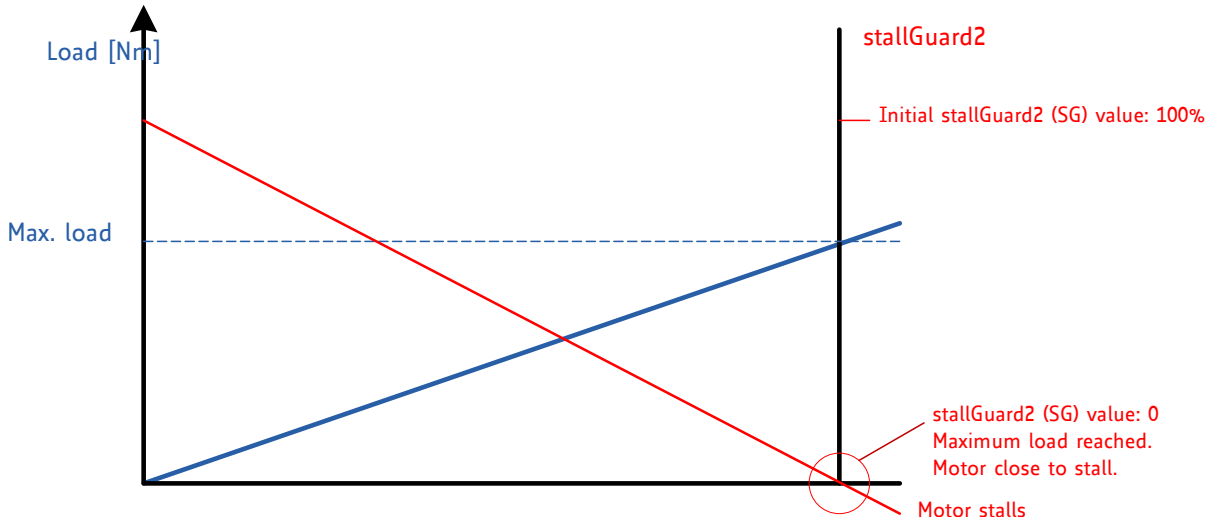


Figure 1: stallGuard2 Load Measurement as a Function of Load

1.4.2 coolStep

coolStep is a load-adaptive automatic current scaling based on the load measurement via stallGuard2 adapting the required current to the load. Energy consumption can be reduced by as much as 75%. coolStep allows substantial energy savings, especially for motors which see varying loads or operate at a high duty cycle. Because a stepper motor application needs to work with a torque reserve of 30% to 50%, even a constant-load application allows significant energy savings because coolStep automatically enables torque reserve when required. Reducing power consumption keeps the ystem cooler, increases motor life, and allows cost reduction.

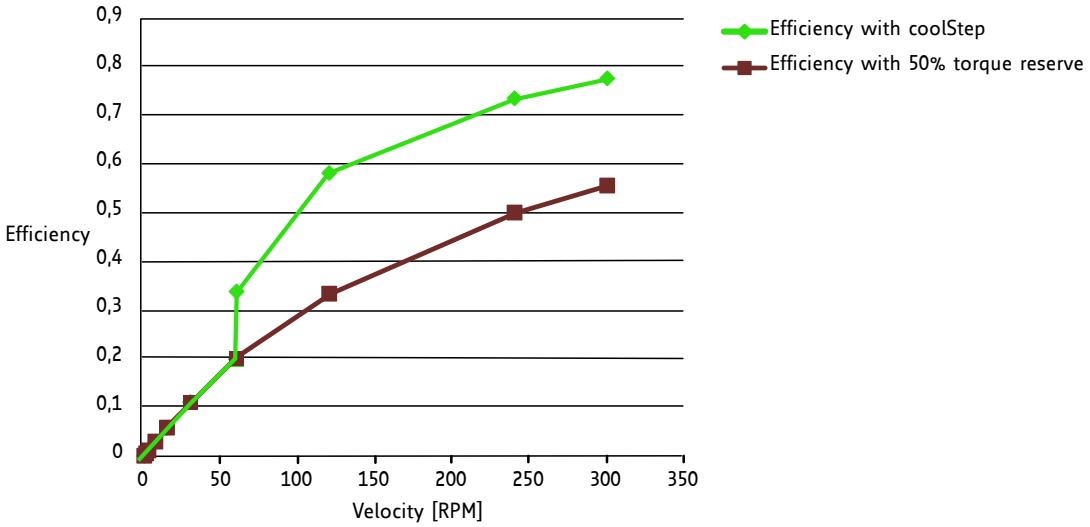


Figure 2: Energy Efficiency Example with coolStep

2 Communication

2.1 Reference Model

The application layer comprises a concept to configure and communicate real-time-data as well as the mechanisms for synchronization between devices. The functionality which the application layer offers to an application is logically divided over different service data objects (SDO) in the application layer. A service object offers a specific functionality and all the related services.

Applications interact by invoking services of a service object in the application layer. To realize these services this object exchanges data via the EtherCAT with peer service object(s) using a protocol.

The application and the application layer interact with service primitives.

Service Primitives	
Primitive	Definition
Request	Issued by the application to the application layer to request a service.
Indication	Issued by the application layer to the application to report an internal event detected by the application layer or indicate that a service is requested.
Response	Issued by the application to the application layer to respond to a previous received indication.
Confirmation	Issued by the application layer to the application to report the result of a previously issued request.

Table 2: Service Primitives

A service type defines the primitives that are exchanged between the application layer and the cooperating applications for a particular service of a service object. Unconfirmed and confirmed services are collectively called remote services.

Service Types	
Type	Definition
Local service	Involves only the local service object. The application issues a request to its local service object that executes the requested service without communicating with peer service object(s).
Unconfirmed service	Involves one or more peer service objects. The application issues a request to its local service object. This request is transferred to the peer service object(s) that each passes it to their application as an indication. The result is not confirmed back.
Confirmed service	Can involve only one peer service object. The application issues a request to its local service object. This request is transferred to the peer service object that passes it to the other application as an indication. The other application issues a response that is transferred to the originating service object that passes it as a confirmation to the requesting application.
Provider initiated service	Involves only the local service object. The service object (being the service provider) detects an event not solicited by a requested service. This event is then indicated to the application.

Table 3: Service Types

2.2 NMT State Machine

The finite state machine (FSM) or simply state machine is a model of behavior composed of a finite number of states, transitions between those states, and actions. It shows which way the logic runs when certain conditions are met.

Starting and resetting the device is controlled via the state machine. The NMT state machine consists of the states shown in figure 3.

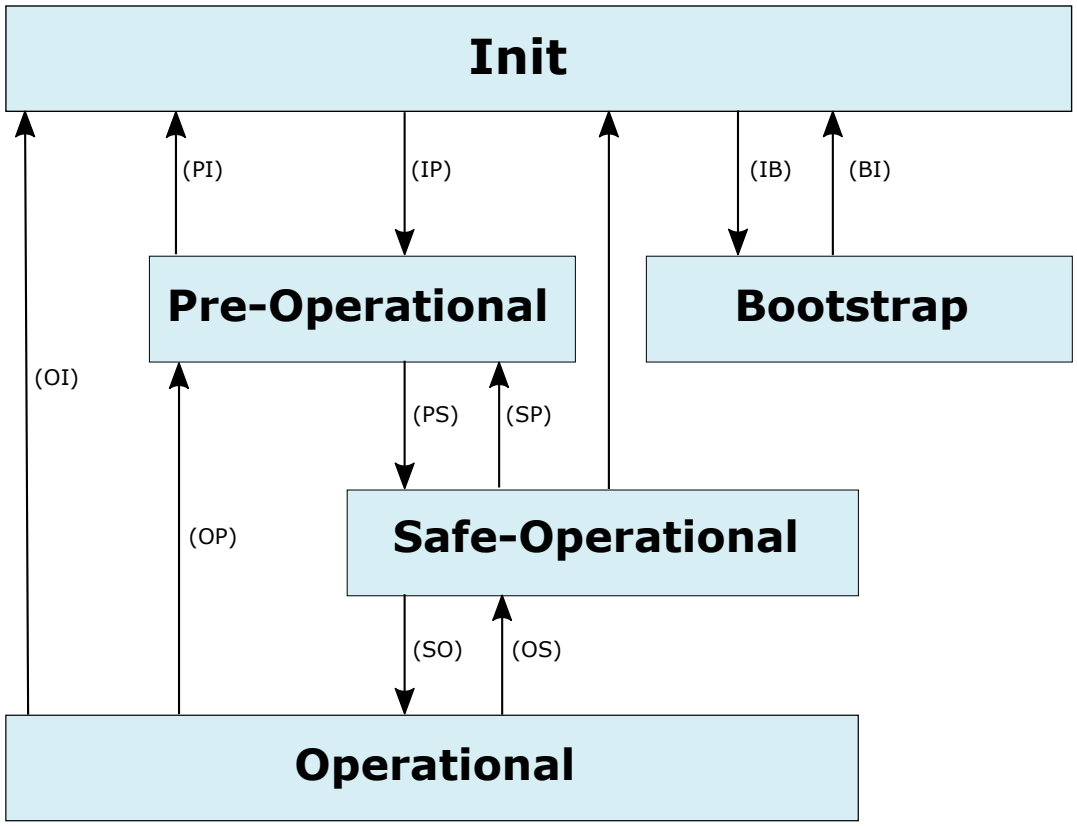


Figure 3: NMT State Machine

After power-on or reset the device enters the Initialization **(INIT)** state.

The master can then switch the device to Pre-Operational **(PRE-OP)** state. In this state, only SDO communication is possible. PDO communication is not possible.

In Safe-Operational **(SAFE-OP)** state, also PDO communication is possible. Inputs can be read, but outputs cannot be switched and the motor cannot be run.

In Operational **(OP)** state, all features of the module can be used. PDO communication is possible, outputs can be switched and the motor can be used. During Operational state the device can use all supported communication objects.

When switching from Operational to Safe-Operational state the motor will be stopped if it has been running. When the EtherCAT connection is lost during Operational state the device will also automatically switch to

Safe-Operational state.

The Bootstrap **(BOOT)** state is used for firmware updates via FoE. Before FoE can be used the device has to be switched to this state.

2.3 Device Model

A CoE device mainly consists of the following parts:

- *Communication:* This function unit provides the communication objects and the appropriate functionality to transport data items via the underlying network structure.
- *Object dictionary:* The object dictionary is a collection of all the data items which have an influence on the behavior of the application objects, the communication objects and the state machine used on this device.
- *Application:* The application comprises the functionality of the device with respect to the interaction with the process environment.

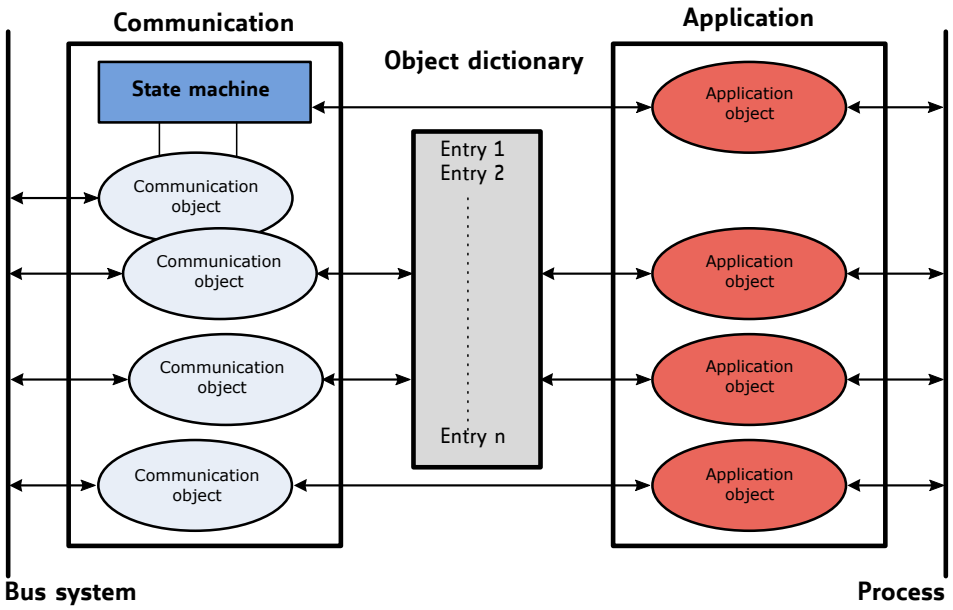


Figure 4: Device Model

2.4 Object Dictionary

The most important part of a device profile is the object dictionary description. The object dictionary is essentially a grouping of objects accessible via the network in an ordered pre-defined fashion. Each object within the dictionary is addressed using a 16-bit index. The overall layout of the standard object dictionary is shown in table 4:

Object Dictionary	
Index	Object
0000_h	Not used.
0001_h – $001F_h$	Static data types.
0020_h – $003F_h$	Complex data types.
0040_h – $005F_h$	Manufacturer specific complex data types.
0060_h – $007F_h$	Device profile specific static data types.
0080_h – $009F_h$	Device profile specific complex data types.
$00A0_h$ – $0FFF_h$	Reserved for further use.
1000_h – $1FFF_h$	Communication profile area.
2000_h – $5FFF_h$	Manufacturer specific profile area.
6000_h – $9FFF_h$	Standardized device profile area.
$A000_h$ – $BFFF_h$	Standardized interface profile area.
$C000_h$ – $FFFF_h$	Reserved for further use.

Table 4: Object Dictionary

The communication profile area at indices 1000_h through $1FFF_h$ contains the communication specific parameters for the CAN network. These entries are common to all devices.

The manufacturer segment at indices 2000_h through $5FFF_h$ contains manufacturer specific objects. These objects control the special features of the Trinamic TMCM-3213 motion control device.

The standardized device profile area at indices 6000_h through $9FFF_h$ contains all data objects common to a class of devices that can be read or written via the network. They describe the device parameters and the device functionality of the device profile.

2.4.1 Object Indices on Multi-Axis Modules

On a multi-axis module like the TMCM-3213 each object in the manufacturer area and each object in the profile specific area is available for each motor. In this manual, only the object indices for motor #0 are shown. The objects for the other motors can be accessed by adding offsets to the object indices:

- Add an offset of $motor\_number \cdot 200_h$ to the index of a manufacturer specifc object to get its index for other motors.
- Add an offset of $motor\_number \cdot 800_h$ to the index of a profile specific object to get its index for other motors.

For example, the control word for motor #1 would be 6840_h (instead of 6040_h for motor #0), and the microstep resolution of motor #1 would be 2200_h for motor #1 (instead of 2000_h for motor #0).

Multi-Axis Object Indices		
Motor	Manufacturer area	Profile area
Motor #0	2000_h – $21FF_h$	6000_h – $67FF_h$
Motor #1	2200_h – $23FF_h$	6800_h – $6FFF_h$
Motor #2	2400_h – $25FF_h$	7000_h – $77FF_h$

Table 5: Multi-Axis Object Indices

3 Communication area

The communication area contains all objects that define the communication parameters of the CoE device according to the EtherCAT standard.

3.1 Detailed object specifications

3.1.1 Object 1000_h: Device Type

This object contains information about the device type. The object 1000_h describes the type of device and its functionality. It is composed of a 16-bit field which describes the device profile that is used and a second 16-bit field which provides additional information about optional functionality of the device.

Object Description			
Index	Name	Object Type	Data Type
1000_h	Device type	Variable	UNSIGNED32

Table 6: Object Description (1000_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	UNSIGNED32	$FFFC0192_h$

Table 7: Entry Description (1000_h)

3.1.2 Object 1001_h: Error Register

This object contains information about the device type. The object 1000_h describes the type of device and its functionality. It is composed of a 16-bit field which describes the device profile that is used and a second 16-bit field which provides additional information about optional functionality of the device.

Object Description			
Index	Name	Object Type	Data Type
1001_h	Error register	Variable	UNSIGNED8

Table 8: Object Description (1001_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	UNSIGNED8	0

Table 9: Entry Description (1001_h)

Error Register Bits	
Bit	Definition
0	Generic error
1	Current
2	Voltage
3	Temperature
4	Communication error
5	Device profile specific
6	Reserved (always 0)
7	Manufacturer specific

Table 10: Error Register Bits

3.1.3 Object 1008_h: Manufacturer Device Name

This object contains the manufacturer device name.

Object Description			
Index	Name	Object Type	Data Type
1008_h	Manufacturer Device Name	Variable	Visible String

Table 11: Object Description (1008_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	—	TMCM-3213

Table 12: Entry Description (1008_h)

3.1.4 Object 1009_h: Manufacturer Hardware Version

This object contains the hardware version description.

Object Description			
Index	Name	Object Type	Data Type
1009_h	Manufacturer Hardware Version	Variable	Visible String

Table 13: Object Description (1009_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	—	Depends on device, e.g. 1.0.

Table 14: Entry Description (1009_h)

3.1.5 Object $100A_h$: Manufacturer Software Version

This object contains the software version description.

Object Description			
Index	Name	Object Type	Data Type
$100A_h$	Manufacturer Software Version	Variable	Visible String

Table 15: Object Description ($100A_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	—	Depends on device, e.g. 1.0.

Table 16: Entry Description ($100A_h$)

3.1.6 Object 1018_h: Identity Object

The object 1018_h contains general information about the device:

- The vendor ID (sub-index 01_h) contains a unique value allocated to each manufacturer. The vendor ID of Trinamic is 286_h.
- The manufacturer specific product code (sub-index 2_h) identifies a specific device version.
- The manufacturer specific revision number (sub-index 3_h) consists of a major revision number and a minor revision number.

Object Description			
Index	Name	Object Type	Data Type
1018_h	Identity object	Record	Identity

Table 17: Object Description (1018_h)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
00_h	Number of entries	ro	no	0...3	3
01_h	Vendor ID	ro	no	UNSIGNED32	0286_h
02_h	Product code	ro	no	UNSIGNED32	3213
03_h	Revision number	ro	no	UNSIGNED32	e.g. 20003_h for version 2.3

Table 18: Entry Description (1018_h)

3.1.7 Object 1600_h: Receive PDO Mapping Parameter

This object contains the mapping parameters for the RPDO the device is able to receive. The sub-index 00_h contains the number of valid entries within the mapping record. This number of entries is also the number of the application variables which shall be received with the corresponding RPDO. The sub-indices from 01_h to the number of entries contain the information about the mapped application variables. These entries describe the PDO contents by their index, sub-index and length.

Object Description			
Index	Name	Object Type	Data Type
1600_h	Receive PDO mapping parameter	RECORD	PDO Mapping

Table 19: Object Description (1600_h)

Entry Description				
Sub-index	Description	Access	Value Range	Default Value
00_h	Number of mapped application objects in PDO	rw	0...9	Index 1600_h: 4
01_h	Mapping entry 1	rw	UNSIGNED32	60400010_h
02_h	Mapping entry 3	rw	UNSIGNED32	$607A0020_h$
03_h	Mapping entry 4	rw	UNSIGNED32	60710010_h
04_h	Mapping entry 5	rw	UNSIGNED32	$60FF0020_h$
05_h	Mapping entry 2	rw	UNSIGNED32	0_h
06_h	Mapping entry 6	rw	UNSIGNED32	0_h
07_h	Mapping entry 7	rw	UNSIGNED32	0_h
08_h	Mapping entry 8	rw	UNSIGNED32	0_h
09_h	Mapping entry 9	rw	UNSIGNED32	0_h

Table 20: Entry Description (1600_h)

3.1.8 Objects $1A00_h$: Transmit PDO Mapping Parameter

This object contains the mapping parameters for the TPDO the device is able to transmit. The sub-index 00_h contains the number of valid entries within the mapping record. This number of entries is also the number of the application variables which shall be transmitted with the corresponding TPDO. The sub-indices from 01_h to the number of entries contain the information about the mapped application variables. These entries describe the PDO contents by their index, sub-index and length.

Object Description			
Index	Name	Object Type	Data Type
$1A00_h$	Transmit PDO mapping parameter	RECORD	PDO Mapping

Table 21: Object Description ($1A00_h$)

Entry Description				
Sub-index	Description	Access	Value Range	Default Value
00_h	Number of mapped aapli-cation objects in PDO	rw	0...9	6
01_h	Mapping entry 1	rw	UNSIGNED32	60410010_h
02_h	Mapping entry 2	rw	UNSIGNED32	60610008_h
03_h	Mapping entry 3	rw	UNSIGNED32	60640020_h
04_h	Mapping entry 4	rw	UNSIGNED32	60770010_h
05_h	Mapping entry 5	rw	UNSIGNED32	$606C0020_h$
06_h	Mapping entry 6	rw	UNSIGNED32	$60FD0020_h$
07_h	Mapping entry 7	rw	UNSIGNED32	0_h
08_h	Mapping entry 8	rw	UNSIGNED32	0_h
09_h	Mapping entry 9	rw	UNSIGNED32	0_h

Table 22: Entry Description ($1A00_h$)

3.1.9 Objects $1C00_h$: Sync Manager Communication Type

This object describes the communication types of the EtherCAT sync managers. The types of the first four synch managers are normally fixed and should not be changed. Sync managers can have the following for communication types:

Sync Manager Communication Types	
Type	Description
1	Mailbox receive
2	Mailbox send
3	Process data input
4	Process data output

Table 23: Sync Manager Communication Types

Object Description			
Index	Name	Object Type	Data Type
$1C00_h$	Sync manager communication type	RECORD	UNSIGNED8

Table 24: Object Description ($1C00_h$)

Entry Description				
Sub-index	Description	Access	Value Range	Default Value
00_h	Number of entries	rw	0...3	4
01_h	Communication type sync manager 1	rw	UNSIGNED8	1
02_h	Communication type sync manager 2	rw	UNSIGNED8	2
03_h	Communictaion type sync manager 3	rw	UNSIGNED8	3
04_h	Communictaion type sync manager 4	rw	UNSIGNED8	4

Table 25: Entry Description ($1C00_h$)

3.1.10 Objects $1C12_h$: Sync Manager 2 PDO Assignment

This object contains the index of the PDO definiton object that is assigned to sync manager 2. Normally, the RPDO objects are assigned to sync manager 2. Under most cicumstances there is no need to change this setting.

Object Description			
Index	Name	Object Type	Data Type
$1C12_h$	Sync manager 2 PDO assignment	RECORD	PDO assignment

Table 26: Object Description ($1C12_h$)

Entry Description				
Sub-index	Description	Access	Value Range	Default Value
00_h	Number of assigned PDOs	rw	0...1	1
01_h	PDO mapping index of assigned RPDO	rw	UNSIGNED16	1600_h

Table 27: Entry Description ($1C12_h$)

3.1.11 Objects $1C13_h$: Sync Manager 3 PDO Assignment

This object contains the index of the PDO definiton object that is assigned to sync manager 3. Normally, the TPDO objects are assigned to sync manager 3. Under most cicumstances there is no need to change this setting.

Object Description			
Index	Name	Object Type	Data Type
$1C13_h$	Sync manager 3 PDO assignment	RECORD	PDO assignment

Table 28: Object Description ($1C13_h$)

Entry Description				
Sub-index	Description	Access	Value Range	Default Value
00_h	Number of assigned PDOs	rw	0...1	1
01_h	PDO mapping index of assigned TPDO	rw	UNSIGNED16	$1A00_h$

Table 29: Entry Description ($1C13_h$)

4 Manufacturer specific area

The manufacturer segment contains manufacturer specific objects. These objects control the special features of the Trinamic Motion Control device TMCM-3213.

Info

This section of the manual only shows the object indices for motor #0. Of course the same objects are also available for the other motors. For the other motors, add an offset of $motor\_number \cdot 200_h$ to the object index. So for example the microstep resolution (object 2000_h for motor #0) can be accessed as object 2200_h for motor #1 and as object 2400_h for motor #2. Please see also section 2.4.1.

Multi-axis Object Indices	
Motor	Object Index Range
Motor #0	2000_h – $21FF_h$
Motor #1	2200_h – $23FF_h$
Motor #2	2400_h – $25FF_h$

Table 30: Multi-axis Object Indices (Manufacturer specific Area)

4.1 Objects related to coolStep

Figure 5 shows an overview of the coolStep related objects for motor #0. Please bear in mind that the figure only shows one example for a drive. There are objects which concern the configuration of the current. Other objects are for velocity regulation and for time adjustment. The coolStep feature is sometimes also called smartEnergy.

The following adjustments have to be made:

- Thresholds for current and velocity have to be identified and set.
- The stallGuard2 feature has to be adjusted and enabled.
- The reduction or increasing of the current in the coolStep area (depending on the load) has to be configured.

coolStep™ adjustment points and thresholds

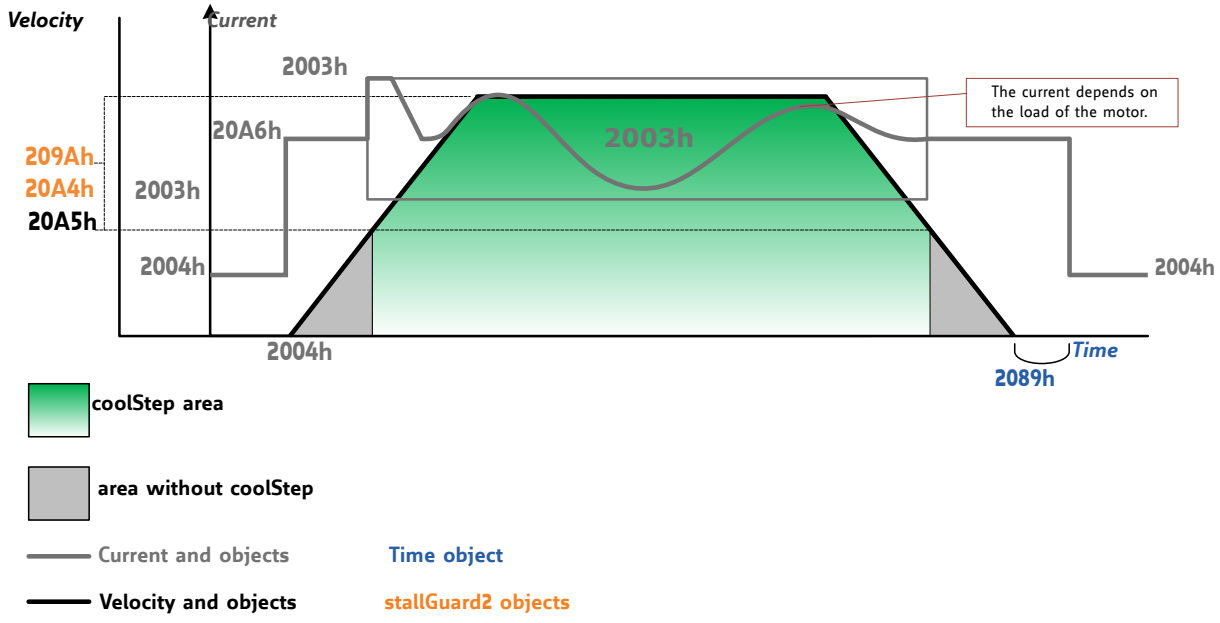


Figure 5: coolStep Adjustment Points and Thresholds

coolStep Adjustment Objects		
Object	Name	Description
2003_h	Absolute maximum current	The maximum value is 255. This value means 100% of the maximum current of the module. The current adjustment is within the range 0...255 and can be adjusted in 32 steps (0...255 divided by eight; step 0 = 0...7, step 1 = 8...15 and so on). The most important motor setting, since too high values might cause motor damage!
2004_h	Standby current	The current limit two seconds after the motor has stopped.
2098_h	smartEnergy current minimum	Sets the lower motor current limit for coolStep operation by scaling the run current (object 2003_h) value. This can be: 0: for 1/2 of the run current 1: for 1/4 of the run current
2099_h	smartEnergy current down step	Sets the speed of current decrement when the stallGuard reading is above the upper threshold. 0: slow decrement 3: fast decrement
$209B_h$	smartEnergy current up step	Sets the current increment step when the stallGuard below the lower threshold. 0: slow increment 3: fast increment / fast reaction to rising load
$209A_h$	smartEnergy hysteresis	Sets the distance between the lower and the upper threshold for stallGuard2 reading. Above the upper threshold the motor current becomes decreased.
$20A4_h$	Stop on stall	Below this speed the motor will not be stopped. Above this speed the motor will stop in case stallGuard2 load value reaches zero.
$20A5_h$	smartEnergy threshold speed.	Above this speed coolStep becomes enabled.
2089_h	Standby delay	Standstill period before the current is changed down to standby current. The standard value is 200 which is 2 seconds.

Table 31: coolStep related Objects