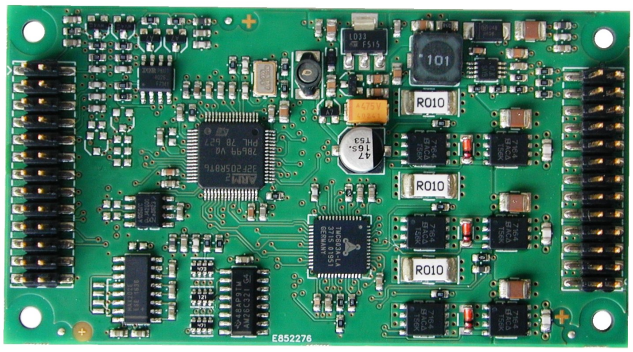
# TMCM-1633 CANopen Firmware Manual

Firmware Version V2.09 | Document Revision V1.00 • 2017-June-28

The TMCM-1633 is a single axis controller module for brushless DC (BLDC) and PMSM motors. It offers field oriented control (FOC) with up-to 10A RMS phase currents at +48V DC supply. Besides hall sensor and incremental ABN encoder interfaces for connection to the motor, digital inputs and outputs can be used. A CAN interface allows communication with a CANopen master.



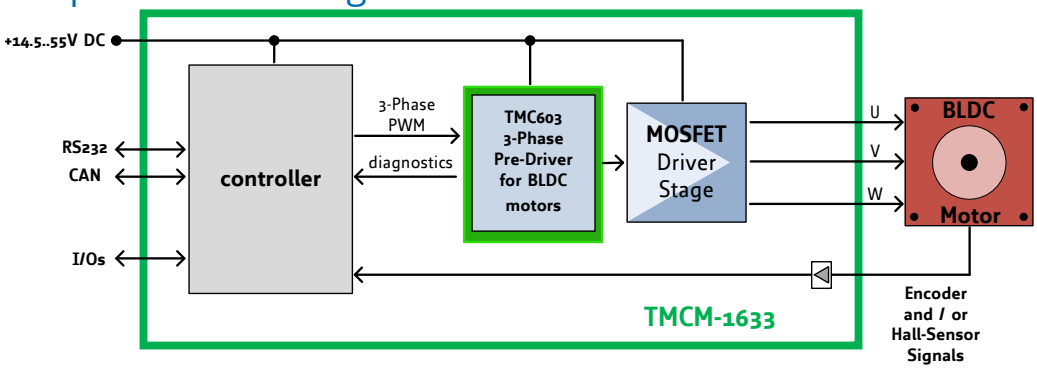
## Features

- Single axis field oriented control for BLDC/PMSM motor
- Hall and ABN encoder support
- +14,5..48V DC supply voltage
- Up to 10A RMS peak motor current
- RS232 & CAN interface
- CANopen CiA 402 drive profile
- Torque, Velocity, and Position control

## Applications

- Life Sciences
- Test & Measurement
- Robotics / Automation

## Simplified Block Diagram






Read entire documentation.

TRINAMIC
MOTION CONTROL

# Contents





# 1 Preface

This document specifies objects and modes of operation of the Trinamic TMCM-1633 BLDC/PMSM motor control module with CANopen firmware. The CANopen firmware is designed to fulfill the CANopen DS402 and DS301 standards. This manual assumes that the reader is already familiar with the basics of the CANopen protocol, defined by the DS301 and DS402 standards of the CAN-CiA.

If necessary, it is always possible to turn the module into a TMCL module by loading the TMCM-1633 TMCL firmware again with the help of the firmware update function of the TMCL-IDE 3.0 and the RS232 interface.

## 1.1 General Features of this CANopen Implementation

**Main Characteristics**

- Communication according to standard CiA-301 V4.1
- CAN bit rate: 20...1000kBit/s
- CAN ID: 11 bit
- Node ID: 1...127 (use vendor specific objects for changing the node ID)
- NMT services: NMT slave

**SDO Communication**

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

**PDO Communication**

- Producer
- Consumer
- RPDOs
  - Axis 0: 1, 2, 3, 4
  - Transmission modes: asynchronous.
  - Dynamic mapping with max. 3 mapping entries.
  - Default mappings: according to CiA-402 for first three PDOs of each axis, manufacturer specific for other PDOs of each axis.
- TPDOs
  - Axis 0: 1, 2, 3, 4
  - Transmission modes: asynchronous, asynchronous with event timer, synchronous.
  - Dynamic mapping with max. 3 mapping entries.
  - Default mappings: according to CiA-402 for first three PDOs of each axis, manufacturer specific for other PDOs of each axis.

**Further Characteristics**

- SYNC: consumer (TPDOs 3 are synchronous PDOs)
- Emergency: producer
- RTR: supported only for node guarding/life guarding
- Heartbeat: consumer and producer

## 1.2 Abbreviations used in this Manual

| Abbreviations | |
|---|---|
| CAN | Controller area network |
| CHGND | chassis ground / earth ground |
| COB | Communication object |
| 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 bootloader and the CANopen firmware itself. Whereas the bootloader 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 RS232 interface of the module.

# 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 CAN Network 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 1.

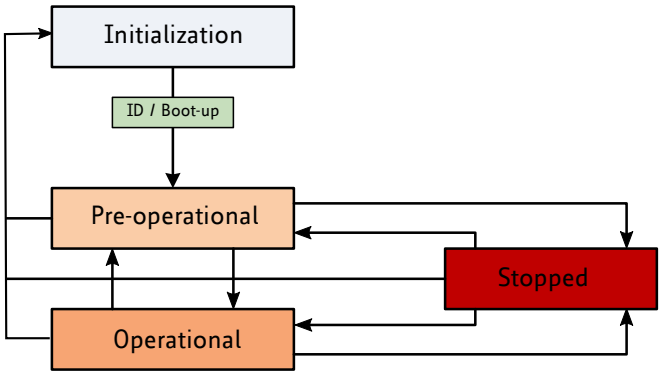



*Figure 1: NMT State Machine*

After power-on or reset the device enters the Initialization state. After the device initialization is finished, the device automatically transits to the **Pre-operational** state and indicates this state transition by sending the boot-up message. This way the device indicates that it is ready to work. A device that stays in Pre-operational state may start to transmit SYNC-, time stamp- or heartbeat message. In contrast to the PDO communication that is disabled in this state, the device can communicate via SDO.

The PDO communication is only possible within the **Operational** state. During Operational state the device can use all supported communication objects.

A device that was switched to the **Stopped** state only reacts on received NMT commands. In addition the device indicates the current NMT state by supporting the error control protocol during Stopped state.

The transitions between states are made by issuing a network management (NMT) communication object to the device. The NMT protocols are used to generate state machine change commands (e.g. to start and stop the device), detect remote device boot-ups and error conditions.

The Heartbeat message of a CANopen device contains the device status of the NMT state machine and is sent cyclically by the CANopen device.

The NMT state machine (or DS301 state machine) is not to be confused with the DS402 state machine. There is only one NMT state machine for the entire device, but for each motor there is a DS402 state machine which controls the motor. There are no links between these state machines, with one exception: When the NMT state machine is being switched to the stopped state, all DS402 state machines that are in OPERATION_ENABLED state will be switch to FAULT state.

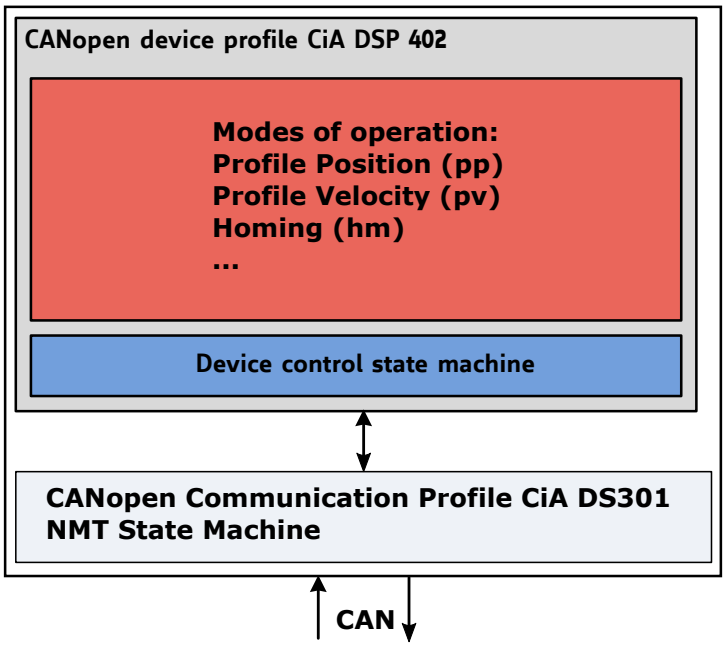



*Figure 2: Communication Architecture*

## 2.3 Device Model

A CANopen 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.

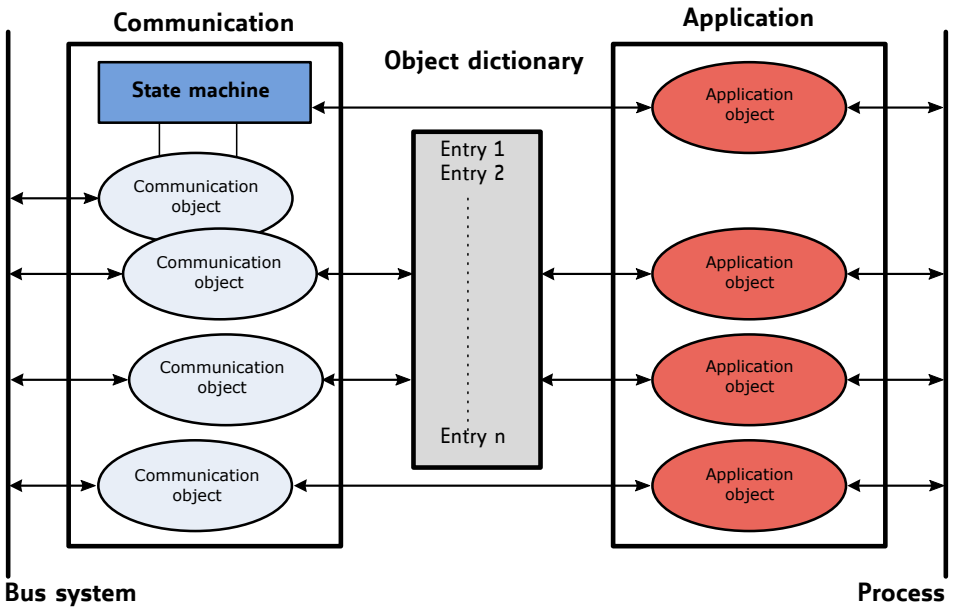



*Figure 3: Device Model*

## 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-1633 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.

# 3 Communication area

The communication area contains all objects that define the communication parameters of the CANopen device according to the DS301 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 5: Object Description ($1000_h$)*

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

*Table 6: 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 7: Object Description ($1001_h$)*

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

*Table 8: 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 9: Error Register Bits*

### 3.1.3 Object $1005_h$: COB-ID SYNC Message

This object defines the COB-ID of the synchronization object (SYNC). Further, it defines whether the module generates the SYNC.

| Value Definition | | |
|---|---|---|
| Bit | Name | Definition |
| 30 | Generate | 0: Device does not generate SYNC message<br>1: Device generates SYNC message |
| 29 | Frame | Not supported, always set to 0. |
| 28...11 | 29 bit ID | Not supported, always set to 0. |
| 10...0 | 11 bit ID | 11 bit COB-ID. |

*Table 10: Value Definition ($1005_h$)*

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1005_h$ | COB-ID SYNC message | Variable | UNSIGNED32 |

*Table 11: Object Description ($1005_h$)*



| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | no | UNSIGNED32 | $80_h$ |

*Table 12: Entry Description ($1005_h$)*

### 3.1.4 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 13: Object Description ($1008_h$)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | ro | no | — | TMCM-1633 |

*Table 14: Entry Description ($1008_h$)*

### 3.1.5 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 15: 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 16: Entry Description ($1009_h$)*

### 3.1.6 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 17: 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 18: Entry Description ($100A_h$)*

### 3.1.7 Object $100C_h$: Guard Time

The objects at index $100C_h$ and $100D_h$ shall indicate the configured guard time respectively the life time factor. The life time factor multiplied with the guard time gives the life time for the life guarding protocol.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $100C_h$ | Guard Time | Variable | UNSIGNED16 |

*Table 19: Object Description ($100C_h$)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | no | UNSIGNED16 | 0 |

*Table 20: Entry Description ($100C_h$)*

### 3.1.8 Object $100D_h$: Life Time Factor

The life time factor multiplied with the guard time gives the life time for the life guarding protocol.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $100D_h$ | Life Time Factor | Variable | UNSIGNED8 |

*Table 21: Object Description ($100D_h$)*

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

*Table 22: Entry Description ($100D_h$)*

### 3.1.9 Object $1010_h$: Store Parameters

This object supports the saving of parameters in non volatile memory. By read access the device provides information about its saving capabilities.

***Note*** This command can only be carried out if the module is in ready to switch on mode.

There are several parameter groups:

- Sub-index $0_h$: contains the largest sub-index that is supported.
- Sub-index $1_h$: saves all parameters.
- Sub-index $2_h$: saves communication parameters $100C_h$, $100D_h$, $1015_h$, $1017_h$, and $1029_h$.
- Sub-index $3_h$: saves device profile parameters.
- Sub-index $4_h$: saves motor 0 parameters.

***Note*** In order to avoid storage of parameters by mistake, storage is only executed when a specific signature is written to the appropriate sub-Index. This signature is "save" ($65766173_h$, see also table 23).

| Save Signature | | | |
|---|---|---|---|
| e | v | a | s |
| $65_h$ | $76_h$ | $61_h$ | $73_h$ |

*Table 23: Save Signature*

On reception of the correct signature in the appropriate sub-index the device stores the parameter and then confirms the SDO transmission (initiate download response). If the storing failed, the device responds with an abort SDO transfer (abort code: $06060000_h$). If a wrong signature is written, the device refuses to store and responds with abort SDO transfer (abort code: $0800002x_h$).

On read access, each sub-index provides information if it is possible to store the parameter group. It reads 1 if yes and 0 if no.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1010_h$ | Store Parameters | Array | UNSIGNED32 |

*Table 24: Object Description ($1010_h$)*

| Entry Description | | | | | |
|---|---|---|---|---|---|
| Sub-index | Description | Access | PDO Mapping | Value Range | Default Value |
| 01h | Save all parameters | rw | no | UNSIGNED32 | — |
| 02h | Save communication parameters | rw | no | UNSIGNED32 | — |
| 03h | Save device profile parameters | rw | no | UNSIGNED32 | — |
| 04h | Save motor axis 0 parameters | rw | no | UNSIGNED32 | — |

*Table 25: Entry Description ($1010_h$)*

### 3.1.10 Object $1011_h$: Restore Parameters

With this object the default values of parameters according to the communication or device profile are restored. By read access the device provides information about its capabilities to restore these values.

***Note*** This command can only be carried out if the module is in ready to switch on mode.

There are several parameter groups:

- Sub-index $0_h$: contains the largest sub-index that is supported.
- Sub-index $1_h$: restores all parameters.
- Sub-index $2_h$: restores communication parameters $100C_h$, $100D_h$, $1015_h$, $1017_h$, and $1029_h$.
- Sub-index $3_h$: restores device profile parameters.
- Sub-index $4_h$: restores motor 0 parameters.

***Note*** In order to avoid restoring the parameters by mistake, restoring is only executed when a specific signature is written to the appropriate sub-Index. This signature is "load" ($64616F6C_h$, see also table 26).

| Load Signature | | | |
|---|---|---|---|
| d | a | o | l |
| $64_h$ | $61_h$ | $6F_h$ | $6C_h$ |

*Table 26: Load Signature*

On reception of the correct signature in the appropriate sub-index the device restores the parameter and then confirms the SDO transmission (initiate download response). If the restoring failed, the device responds with an abort SDO transfer (abort code: $06060000_h$). If a wrong signature is written, the device refuses to restore and responds with abort SDO transfer (abort code: $0800002x_h$).

On read access, each sub-index provides information if it is possible to restore the parameter group. It reads 1 if yes and 0 if no.

After the default values have been restored they will become active after the next rest or power cycle of the TMCM-1633.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1011_h$ | Restore parameters | Array | UNSIGNED32 |

*Table 27: Object Description ($1011_h$)*

| Entry Description | | | | | |
|---|---|---|---|---|---|
| Sub-index | Description | Access | PDO Mapping | Value Range | Default Value |
| 01h | Restore all parameters | rw | no | UNSIGNED32 | — |
| 02h | Restore communication parameters | rw | no | UNSIGNED32 | — |
| 03h | Restore device profile parameters | rw | no | UNSIGNED32 | — |
| 04h | Restore motor axis 0 parameters | rw | no | UNSIGNED32 | — |

*Table 28: Entry Description ($1011_h$)*

### 3.1.11 Object $1014_h$: COB-ID Emergency Object

This object defines the COB-ID of the emergency object (EMCY).

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1014_h$ | COB-ID emergency object | Variable | UNSIGNED32 |

*Table 29: Object Description ($1014_h$)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | no | UNSIGNED32 | $80_h$ + Node ID |

*Table 30: Entry Description ($1014_h$)*



### 3.1.12 Object $1015_h$: Inhibit Time EMCY

The inhibit time for the EMCY message can be adjusted via this entry. The time has to be a multiple of $100\mu s$.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1015_h$ | COB-ID emergency object | Variable | UNSIGNED16 |

*Table 31: Object Description ($1015_h$)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | no | UNSIGNED16 | 0 |

*Table 32: Entry Description ($1015_h$)*

### 3.1.13 Object $1016_h$: Consumer Heartbeat Time

The consumer heartbeat time defines the expected heartbeat cycle time and thus has to be higher than the corresponding producer heartbeat time configured on the module producing this heartbeat. The monitoring starts after the reception of the first heartbeat. If the consumer heartbeat time is 0 the corresponding entry is not used. The time has to be a multiple of 1ms.

| Value Definition | | |
|---|---|---|
| Bits | Name | Definition |
| 31...24 | Reserved | — |
| 23...16 | Node ID | Heartbeat Producer Node ID |
| 15...0 | Heartbeat time | Time in 1ms |

*Table 33: Value Definition ($1016_h$)*

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1016_h$ | Consumer heartbeat time | Variable | UNSIGNED16 |

*Table 34: Object Description ($1016_h$)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | no | UNSIGNED16 | 0 |

*Table 35: Entry Description ($1016_h$)*

### 3.1.14 Object $1017_h$: Producer Heartbeat Time

The producer heartbeat time defines the cycle time of the heartbeat. The producer heartbeat time is 0 if it is not used. The time has to be a multiple of 1ms.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1017_h$ | Producer heartbeat time | Variable | UNSIGNED16 |

*Table 36: Object Description ($1017_h$)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | no | UNSIGNED16 | 0 |

*Table 37: Entry Description ($1017_h$)*

### 3.1.15 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 38: 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 | 1633 |
| $03_h$ | Revision number | ro | no | UNSIGNED32 | e.g. $20003_h$ for version 2.3 |

*Table 39: Entry Description ($1018_h$)*

### 3.1.16 Object $1029_h$: Error Behaviour

If a device failure is detected in operational state, the device can be configured to enter alternatively the stopped state or remain in the current state in case of a device failure. Device failures include the following errors:

- Communication error
- Application error

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1029_h$ | Error behaviour | Array | UNSIGNED8 |

*Table 40: Object Description ($1029_h$)*

| Entry Description | | | | | |
|---|---|---|---|---|---|
| Sub-index | Description | Access | PDO Mapping | Value Range | Default Value |
| $00_h$ | Number of error classes | ro | no | — | 2 |
| $01_h$ | Communication error | rw | no | UNSIGNED8 | 0 (enter stopped state) |
| $02_h$ | Application error | rw | no | UNSIGNED8 | 1 (remain in current state) |

*Table 41: Entry Description ($1029_h$)*

### 3.1.17 Objects $1400_h$ – $1403_h$: Receive PDO Communication Parameter

This object contains the communication parameters for the RPDOs which the device is able to receive. The sub-index $00_h$ contains the number of valid entries within the communication record. Its value normally is 2, as this object consists of two other entries.

Sub-index $01_h$ contains the COB-ID used by this PDO (in bits 10...0). Bit 30 (RTR bit) defines if this PDO uses RTRs. As RTRs are not supported for PDOs by this CANopen implementation, this bit must always be set in order to turn off RTR support for this PDO. Bit 31 defines if this PDO is active or not. If this bit is set, the PDO is inactive, and if this bit is clear, the PDO is active. Before making any changes to a PDO definition, set this bit to inactivate the PDO.

Sub-Index $02_h$ contains the transmission type of the RPDO. This can be $FF_h$ or $FE_h$ for event-driven, or $00_h$ for synchronous.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1400_h$ – $1403_h$ | Receive PDO parameter | RECORD | RPDO CommPar |
| $1400_h$ | RPDO 1 | RECORD | RPDO CommPar |
| $1401_h$ | RPDO 2 | RECORD | RPDO CommPar |
| $1402_h$ | RPDO 3 | RECORD | RPDO CommPar |
| $1403_h$ | RPDO 4 | RECORD | RPDO CommPar |

*Table 42: Object Description ($1400_h$)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Description | Access | Value Range | Default Value |
| $00_h$ | Largest sub-index supported | ro | 2 | 2 |
| $01_h$ | COB-ID used by PDO | rw | UNSIGNED32 | Index $1400_h$: $200_h$ + Node-ID<br>Index $1401_h$: $300_h$ + Node-ID<br>Index $1402_h$: $400_h$ + Node-ID<br>Index $1403_h$: $500_h$ + Node-ID |
| $02_h$ | Transmission type | rw | UNSIGNED8 | Index $1400_h$: $FF_h$<br>Index $1401_h$: $FF_h$<br>Index $1402_h$: $FF_h$<br>Index $1403_h$: $FE_h$ |

*Table 43: Entry Description ($1400_h$)*

### 3.1.18 Objects $1600_h$ – $1603_h$: Receive PDO Mapping Parameter

These objects contain the mapping parameters for the RPDOs 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 |
| $1600_h$ – $1603_h$ | Receive PDO mapping parameter | RECORD | PDO Mapping |
| $1600_h$ | RPDO 1 | RECORD | PDO Mapping |
| $1601_h$ | RPDO 2 | RECORD | PDO Mapping |
| $1602_h$ | RPDO 3 | RECORD | PDO Mapping |
| $1603_h$ | RPDO 4 | RECORD | PDO Mapping |

*Table 44: 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...3 | Index $1600_h$: 1<br>Index $1601_h$: 2<br>Index $1602_h$: 2<br>Index $1603_h$: 2 |
| $01_h$ | Mapping entry 1 | rw | UNSIGNED32 | Index $1600_h$: $60400010_h$<br>Index $1601_h$: $60400010_h$<br>Index $1602_h$: $60400010_h$<br>Index $1603_h$: $60400010_h$ |
| $02_h$ | Mapping entry 2 | rw | UNSIGNED32 | Index $1600_h$: 0<br>Index $1601_h$: $60600008_h$<br>Index $1602_h$: $607A0020_h$<br>Index $1603_h$: $60FF0020_h$ |
| $03_h$ | Mapping entry 3 | rw | UNSIGNED32 | Index $1600_h$: $0_h$<br>Index $1601_h$: $0_h$<br>Index $1602_h$: $0_h$<br>Index $1603_h$: $0_h$ |

*Table 45: Entry Description ($1600_h$)*

Before making changes to PDO definitions, first mark the PDO as inactive by setting bit 31 of its COB-ID (see section 3.1.17). Then, set its number of mapped PDO entries to zero (sub-index 0 of the appropriate PDO mapping object). Now, the mapppings themself can be changed. After that, set the number of map objects to the desired value, and finally activate the PDO by clearing bit 31 of its COB-ID.

### 3.1.19 Objects $1800_h$ – $1803_h$: Transmit PDO Communication Parameter

This object contains the communication parameters for the TPDOs which the device is able to transmit. The sub-index $00_h$ contains the number of valid entries within the communication record. Its value normally is 5, as this object consists of five other entries.

Sub-index $01_h$ contains the COB-ID used by this PDO (in bits 10...0). Bit 30 (RTR bit) defines if this PDO uses RTRs. As RTRs are not supported for PDOs by this CANopen implementation, this bit must always be set in order to turn off RTR support for this PDO. Bit 31 defines if this PDO is active or not. If this bit is set, the PDO is inactive, and if this bit is clear, the PDO is active. Before making any changes to a PDO

definition, set this bit to inactivate the PDO.

Sub-index $02_h$ contains the transmission type of the RPDO. This can be $FF_h$ or $FE_h$ for event-driven, or $00_h$ or $01_h$ for synchronous.

Sub-index $03_h$ contains the inhibit time, given in milliseconds. After a TPDO has been sent, it will not be sent again before the inhibit time has elapsed.

Sub-index $04_h$ is not used.

Sub-index $05_h$ contains the event timer value in milliseconds. When this is set to a value greater than 0 the TPDO will be sent repeatedly each time the event timer has elapsed. For example, when this value is set to 250, the TPDO will be sent every 250ms.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $1800_h$ – $1803_h$ | Transmit PDO communication parameter | RECORD | TPDO CommPar |
| $1800_h$ | TPDO 1 | RECORD | TPDO CommPar |
| $1801_h$ | TPDO 2 | RECORD | TPDO CommPar |
| $1802_h$ | TPDO 3 | RECORD | TPDO CommPar |
| $1803_h$ | TPDO 4 | RECORD | TPDO CommPar |

*Table 46: Object Description ($1800_h$)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Description | Access | Value Range | Default Value |
| $00_h$ | Largest sub-index supported | ro | 5 | 5 |
| $01_h$ | COB-ID | rw | UNSIGNED32 | Index $1800_h$: $180_h$ + Node-ID<br>Index $1801_h$: $280_h$ + Node-ID<br>Index $1802_h$: $380_h$ + Node-ID<br>Index $1803_h$: $480_h$ + Node-ID |
| $02_h$ | Transmission type | rw | UNSIGNED8 | Index $1800_h$: $FF_h$<br>Index $1801_h$: $FF_h$<br>Index $1802_h$: $01_h$<br>Index $1803_h$: $01_h$ |
| $03_h$ | Inhibit time | rw | UNSIGNED16 | 0 |
| $04_h$ | Compatibility entry | ro | UNSIGNED8 | 0 |
| $05_h$ | Event timer | rw | UNSIGNED16 | 0 |

*Table 47: Entry Description ($1800_h$)*

### 3.1.20 Objects $1A00_h$ – $1A03_h$: Transmit PDO Mapping Parameter

These objects contain the mapping parameters for the TPDOs 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 |
| $1A00_h$ – $1A03_h$ | Transmit PDO mapping parameter | RECORD | PDO Mapping |
| $1A00_h$ | TPDO 1 | RECORD | PDO Mapping |
| $1A01_h$ | TPDO 2 | RECORD | PDO Mapping |
| $1A02_h$ | TPDO 3 | RECORD | PDO Mapping |
| $1A03_h$ | TPDO 4 | RECORD | PDO Mapping |

*Table 48: 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...3 | Index $1A00_h$: 1<br>Index $1A01_h$: 2<br>Index $1A02_h$: 2<br>Index $1A03_h$: 2 |
| $01_h$ | Mapping entry 1 | rw | UNSIGNED32 | Index $1A00_h$: $60410010_h$<br>Index $1A01_h$: $60410010_h$<br>Index $1A02_h$: $60410010_h$<br>Index $1A03_h$: $60410010_h$ |
| $02_h$ | Mapping entry 2 | rw | UNSIGNED32 | Index $1A00_h$: 0<br>Index $1A01_h$: $60610008_h$<br>Index $1A02_h$: $60640020_h$<br>Index $1A03_h$: $606C0020_h$ |
| $03_h$ | Mapping entry 3 | rw | UNSIGNED32 | Index $1A00_h$: $0_h$<br>Index $1A01_h$: $0_h$<br>Index $1A02_h$: $0_h$<br>Index $1A03_h$: $0_h$ |

*Table 49: Entry Description ($1A00_h$)*

Before making changes to PDO definitions, first mark the PDO as inactive by setting bit 31 of its COB-ID (see section 3.1.19). Then, set its number of mapped PDO entries to zero (sub-index 0 of the appropriate PDO mapping object). Now, the mapppings themself can be changed. After that, set the number of map objects to the desired value, and finally activate the PDO by clearing bit 31 of its COB-ID.

# 4 Manufacturer specific area

The manufacturer segment contains manufacturer specific objects. These objects control the special features of the Trinamic Motion Control device TMCM-1633.

## 4.1 Detailed object specifications

### 4.1.1 Object $2005_h$: Limit Switches

This object defines which limit switches are to be used. Bit 0 stands for the left and bit 1 stands for the right limit switch. If a bit is set, the corresponding limit switch will not be used. So this object has to be set to the value 3 if limit switches are not connected. The object can only be written when the drive is in the SWITCHED_ON_DISABLED state (but is always readable).
The limit switches can also be inverted using bit 2 and bit 3:

- Bit 2 inverts the left limit switch
- Bit 3 inverts the right limit switch

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2005_h$ | Limit switches | Variable | UNSIGNED32 |

*Table 50: Object Description ($2005_h$)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | no | 0...63 | 0 |

*Table 51: Entry Description ($2005_h$)*

| Bit Definitions | |
|---|---|
| Bit | Definition |
| 0 | Left limit switch deactivated, if set. |
| 1 | Right limit switch deactivated, if set. |
| 2 | Left limit switch inverted, if set. |
| 3 | Right limit switch inverted, if set. |
| 4 | Home switch deactivated, if set. |
| 5 | Home switch inverted, if set. |

*Table 52: Bit Definitions ($2005_h$)*

### 4.1.2 Object 200D$_h$: Status Flags

This object provides information about the actual module status flags. (0: not active, 1: active).

This object is organized bit-wise. The bits have the following meaning:

Bit 0: OVERCURRENT
Bit 1: UNDERVOLTAGE
Bit 2: OVERVOLTAGE
Bit 3: OVERTEMPERATURE
Bit 4: MOTORHALTED
Bit 5: HALLERROR
Bit 6: DRIVER_ERROR
Bit 7: INIT_ERROR
Bit 8: STOP_MODE
Bit 9: VELOCITY_MODE
Bit 10: POSITION_MODE
Bit 11: TORQUE_MODE
Bit 12: EMERGENCYSTOP
Bit 13: FREERUNNING
Bit 14: POSITION_END
Bit 15: MODULE_INITIALIZED
Bit 16: unused
Bit 17: IIT_EXCEEDED

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| 200D$_h$ | Status Flags | Variable | UNSIGNED32 |

*Table 53: Object Description (200D$_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Status Flags | no | 0 | 4294967295 | 0 | | R |

*Table 54: Entry Description (200D$_h$)*

### 4.1.3 Object 200E$_h$: Supply Voltage

The actual supply voltage.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| 200E$_h$ | Supply Voltage | Variable | UNSIGNED32 |

*Table 55: Object Description (200E$_h$)*



| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Supply Voltage | no | 0 | 1000 | 300 | [100mV] | R |

*Table 56: Entry Description ($200E_h$)*

### 4.1.4 Object $200F_h$: Driver Temperatur

The actual temperature of the motor driver.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $200F_h$ | Driver Temperatur | Variable | SIGNED32 |

*Table 57: Object Description ($200F_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Driver Temperature | no | -20 | 150 | 0 | [degree] | R |

*Table 58: Entry Description ($200F_h$)*

### 4.1.5 Object $2010_h$: Motor Settings

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2010_h$ | Motor Settings | Variable | Record |

*Table 59: Object Description ($2010_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | MotorPoles | no | 2 | 254 | 8 | | RW |

*Table 60: Entry Description ($2010_h$)*

### 4.1.6 Object $2020_h$: Limits

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2020_h$ | Limits | Variable | Record |

*Table 61: Object Description ($2020_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | MaxTorque | no | 0 | 15000 | 4000 | [mA] (peak) | RW |
| 2 | MaxVelocity | no | 0 | 200000 | 4000 | [rpm] | RW |
| 3 | MaxAcceleration | no | 0 | 100000 | 2000 | [rpm/s] | RW |

*Table 62: Entry Description ($2020_h$)*

### 4.1.7 Object $2030_h$: Torque Mode Settings

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2030_h$ | Torque Mode Settings | Variable | Record |

*Table 63: Object Description ($2030_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | ActualCurrent | no | -2147483648 | 2147483647 | 0 | [mA] (peak) | R |
| 2 | TargetCurrent | no | -15000 | 15000 | 0 | [mA] (peak) | R |
| 3 | RampTargetCurrent | no | -15000 | 15000 | 0 | [mA] (peak) | R |
| 4 | P_Parameter | no | 0 | 65535 | 0 | | RW |
| 5 | I_Parameter | no | 0 | 65535 | 0 | | RW |
| 6 | PI_Torque_Error | no | -2147483648 | 2147483647 | 0 | [mA] | R |
| 7 | PI_Torque_Error_Sum | no | -2147483648 | 2147483647 | 0 | | R |
| 8 | PI_Flux_Error | no | -2147483648 | 2147483647 | 0 | [mA] | R |
| 9 | PI_Flux_Error_Sum | no | -2147483648 | 2147483647 | 0 | | R |

*Table 64: Entry Description ($2030_h$)*



### 4.1.8 Object $2040_h$: Velocity Mode Settings

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2040_h$ | Velocity Mode Settings | Variable | Record |

*Table 65: Object Description ($2040_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | ActualVelocity | no | -2147483648 | 2147483647 | 0 | [rpm] | R |
| 2 | TargetVelocity | no | -200000 | 200000 | 0 | [rpm] | R |
| 3 | RampTargetVelocity | no | -2147483648 | 2147483647 | 0 | [rpm] | R |
| 4 | MotorHaltedVelocity | no | 0 | 200000 | 5 | [rpm] | RW |
| 5 | P_Parameter | no | 0 | 65535 | 0 | | RW |
| 6 | I_Parameter | no | 0 | 65535 | 0 | | RW |
| 7 | PI_Velocity_Error | no | -2147483648 | 2147483647 | 0 | [rpm] | R |
| 8 | PI_Velocity_Error_Sum | no | -2147483648 | 2147483647 | 0 | | R |

*Table 66: Entry Description ($2040_h$)*

### 4.1.9 Object $2050_h$: Position Mode Settings

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2050_h$ | Position Mode Settings | Variable | Record |

*Table 67: Object Description ($2050_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | ActualPosition | no | -2147483648 | 2147483647 | 0 | | RW |
| 2 | TargetPosition | no | -2147483648 | 2147483647 | 0 | | R |
| 3 | RampTargetPosition | no | -2147483648 | 2147483647 | 0 | | R |
| 4 | P_Parameter | no | 0 | 65535 | 0 | | RW |
| 5 | PI_Position_Error | no | -2147483648 | 2147483647 | 0 | | R |
| 6 | TargetReachedVelocity | no | 0 | 200000 | 500 | [rpm] | RW |
| 7 | TargetReachedDistance | no | 0 | 100000 | 5 | | RW |

*Table 68: Entry Description ($2050_h$)*

### 4.1.10 Object $2055_h$: Commutation Mode

Select a commutation mode that fits best to your motor's sensors.

6 : FOC (hall sensor)
7 : FOC (encoder)
8 : FOC (controlled)

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2055_h$ | Commutation Mode | Variable | Record |

*Table 69: Object Description ($2055_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Commutation Mode | no | 6 | 8 | 6 | | RW |

*Table 70: Entry Description ($2055_h$)*

### 4.1.11 Object $2056_h$: Velocity Ramp Mode

An activated ramp allows a defined acceleration for velocity and position mode.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2056_h$ | Velocity Ramp Mode | Variable | UNSIGNED8 |

*Table 71: Object Description ($2056_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Velocity Ramp Mode | no | 0 | 1 | 1 | | RW |

*Table 72: Entry Description ($2056_h$)*

### 4.1.12 Object $2060_h$: Open Loop Settings

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2060_h$ | Open Loop Settings | Variable | Record |

*Table 73: Object Description ($2060_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | ActualAngle | no | -32768 | 32767 | 0 | | R |
| 2 | OpenLoopCurrent | no | 0 | 15000 | 1500 | [mA] (peak) | RW |

*Table 74: Entry Description ($2060_h$)*

### 4.1.13 Object $2070_h$: Hall Sensor Settings

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2070_h$ | Hall Sensor Settings | Variable | Record |

*Table 75: Object Description ($2070_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | ActualAngle | no | -32768 | 32767 | 0 | | R |
| 2 | Inversion | no | 0 | 1 | 0 | | RW |
| 3 | Interpolation | no | 0 | 1 | 0 | | RW |

*Table 76: Entry Description ($2070_h$)*



### 4.1.14 Object $2080_h$: ABN Encoder Settings

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2080_h$ | ABN Encoder Settings | Variable | Record |

*Table 77: Object Description ($2080_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | ActualAngle | no | -32768 | 32767 | 0 | | R |
| 2 | StepsPerRotation | no | 0 | 65535 | 4000 | | RW |
| 3 | Offset | no | 0 | 65535 | 0 | | RW |
| 4 | Direction | no | 0 | 1 | 0 | | RW |
| 5 | InitMode | no | 0 | 2 | 1 | | RW |
| 6 | InitDelay | no | 0 | 10000 | 1000 | [ms] | RW |
| 7 | InitVelocity | no | -200000 | 200000 | 100 | [rpm] | RW |

*Table 78: Entry Description ($2080_h$)*

### 4.1.15 Object $2100_h$: Home Offset Display

This object shows the home offset. The value is given in encoder or hall increments.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2100_h$ | Home Offset Display | Variable | SIGNED32 |

*Table 79: Object Description ($2100_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Home Offset Display | no | -2147483648 | 2147483647 | 0 | | R |

*Table 80: Entry Description ($2100_h$)*

### 4.1.16 Object $2702_h$: Digital Inputs

Bit0: Left limit switch status



Bit1: Right limit switch status

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2702_h$ | Digital Inputs | Variable | UNSIGNED32 |

*Table 81: Object Description ($2702_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Digital Inputs | no | 0 | 3 | 0 | | R |

*Table 82: Entry Description ($2702_h$)*

### 4.1.17 Object $2704_h$: CAN Bit Rate

With this object it is possible to change the CAN bit rate.

To do this, first write the new value to this object. Then, store the new setting by writing the save signature to object 2706h. After that, reset the module. The new setting then becomes active.

(Available bit rates: 20, 50, 100, 125, 250, 500, 800, 1000)

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2704_h$ | CAN Bit Rate | Variable | UNSIGNED16 |

*Table 83: Object Description ($2704_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | CAN Bit Rate | no | 20 | 1000 | 1000 | | RW |

*Table 84: Entry Description ($2704_h$)*

### 4.1.18 Object $2705_h$: Node ID

On modules that do not have address switches the node ID can be selected using this object.

On modules with address switches the node ID is normally selected using the address switches.

To change the node ID, first write the new node ID to this object. Then, store the new setting by writing the save signature to object 2706h. After that, reset the module. The new setting then becomes active.



| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2705_h$ | Node ID | Variable | UNSIGNED8 |

*Table 85: Object Description ($2705_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Node ID | no | 1 | 127 | 1 | | RW |

*Table 86: Entry Description ($2705_h$)*

### 4.1.19 Object $2706_h$: Store

Writing the save signature to this object permanently saves changes made to objects 2704h and 2705h. The save signature is 65766173h.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2706_h$ | Store | Variable | UNSIGNED32 |

*Table 87: Object Description ($2706_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Store | no | 0 | 4294967295 | 0 | | RW |

*Table 88: Entry Description ($2706_h$)*

### 4.1.20 Object $2707_h$: CAN Bit Rate Load

This object shows the selected CAN bit rate.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2707_h$ | CAN Bit Rate Load | Variable | UNSIGNED8 |

*Table 89: Object Description ($2707_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | CAN Bit Rate Load | no | 20 | 1000 | 1000 | | R |

*Table 90: Entry Description ($2707_h$)*

### 4.1.21 Object $2708_h$: Node ID Load

This object shows the selected node ID.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2708_h$ | Node ID Load | Variable | UNSIGNED8 |

*Table 91: Object Description ($2708_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Node ID Load | no | 1 | 127 | 1 | | R |

*Table 92: Entry Description ($2708_h$)*

### 4.1.22 Object $270E_h$: Analog Inputs

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $270E_h$ | Analog Inputs | Variable | Record |

*Table 93: Object Description ($270E_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | ADC_IN_0 | no | 0 | 4095 | 0 | | R |
| 2 | ADC_IN_1 | no | 0 | 4095 | 0 | | R |
| 3 | ADC_phase_A | no | 0 | 4095 | 0 | | R |
| 4 | ADC_phase_B | no | 0 | 4095 | 0 | | R |
| 5 | ADC_phase_C | no | 0 | 4095 | 0 | | R |
| 6 | ADC_VSupply | no | 0 | 4095 | 0 | | R |
| 7 | ADC_Temp | no | 0 | 4095 | 0 | | R |

*Table 94: Entry Description ($270E_h$)*

# 5 Profile specific area

The profile segment contains CiA-402 standard motion control objects. These objects control the motion control functions of the TMCM-1633. Since it is not possible to operate the modes in parallel, the user is able to activate the required function by selecting a mode of operation. The control device writes to the modes of operation object in order to select the operation mode. The drive device provides the modes of operation display object to indicate the actual activated operation mode. Controlword, statusword, and set-points are used mode-specific. This implies the responsibility of the control device to avoid inconsistencies and erroneous behavior.

The following operating modes (selectable via object $6060_h$, please see 5.1.6) are implemented on the TMCM-1633:

- Profile position mode (pp)
- Profile velocity mode (pv)
- Cyclic torque mode (cst)
- Homing mode (hm)

## 5.1 Detailed object specifications

### 5.1.1 Object $605A_h$: Quick Stop Option Code

This object indicates what action is performed when the quick stop function is executed. The slow down ramp is the deceleration value of the used mode of operation. The following quick stop option codes are supported in the current version of the CANopen firmware:

1: Slow down on slow down ramp and transit into switch on disabled
2: Slow down on quick stop ramp and transit into switch on disabled
5: Slow down on slow down ramp and stay in quick stop active)
6: Slow down on quick stop ramp and stay in quick stop active

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $605A_h$ | Quick Stop Option Code | Variable | SIGNED16 |

*Table 95: Object Description ($605A_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Quick Stop Option Code | no | 1 | 6 | 0 | | RW |

*Table 96: Entry Description ($605A_h$)*

### 5.1.2 Object $605B_h$: Shutdown Option Code

This object indicates what action is performed if there is a transition from operation enabled state to ready to switch on state. The shutdown option code always has the value 0 as only this is supported.

0: Disable drive function (switch off the power stage)

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $605B_h$ | Shutdown Option Code | Variable | SIGNED16 |

*Table 97: Object Description ($605B_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Shutdown Option Code | no | 0 | 0 | 0 | | RW |

*Table 98: Entry Description ($605B_h$)*

### 5.1.3 Object $605C_h$: Disable Operation Option Code

This object indicates what action is performed if there is a transition from operation enabled state to switched on state. The disable operation option code always has the value 1 as only this is supported. The slow down ramp is the deceleration value of the used mode of operation.

1: Slow down on slow down ramp



| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $605C_h$ | Disable Operation Option Code | Variable | SIGNED16 |

*Table 99: Object Description ($605C_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Disable Operation Option Code | no | 1 | 1 | 1 | | RW |

*Table 100: Entry Description ($605C_h$)*

### 5.1.4 Object $605D_h$: Halt Option Code

This object indicates what action is performed when the halt function is executed. The slow down ramp is the deceleration value of the used mode of operation. The halt option code always has the value 1 as only this is supported.

1: Slow down on slow down ramp and stay in operation enabled

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $605D_h$ | Halt Option Code | Variable | SIGNED16 |

*Table 101: Object Description ($605D_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Halt Option Code | no | 1 | 1 | 1 | | RW |

*Table 102: Entry Description ($605D_h$)*

### 5.1.5 Object $605E_h$: Fault Reaction Option Code

This object indicates what action is performed when fault is detected in the power drive system. The slow down ramp is the deceleration value of the used mode of operation. The fault reaction option code always has the value 2 as only this is supported.

2: Slow down on quick stop ramp

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $605E_h$ | Fault Reaction Option Code | Variable | SIGNED16 |

*Table 103: Object Description ($605E_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Fault Reaction Option Code | no | 2 | 2 | 2 | | RW |

*Table 104: Entry Description ($605E_h$)*

### 5.1.6 Object $6060_h$: Modes of Operation

This object indicates the requested operation mode. Supported operating modes are:

0: No mode
1: Profile position mode (pp)
3: Profile velocity mode (pv)
6: Homing mode (hm)
10: Cyclic torque mode (cst)

The motor will not run when the operating mode is set to 0. It will be stopped when the motor is running in one of the supported operating modes and the operating mode is then switched to 0.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6060_h$ | Modes of Operation | Variable | SIGNED8 |

*Table 105: Object Description ($6060_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Modes of Operation | no | 0 | 10 | 0 | | RW |

*Table 106: Entry Description ($6060_h$)*

### 5.1.7 Object $6061_h$: Modes of Operation Display

This object shows the operating mode that is currently set.

0: No mode



1: Profile position mode (pp)
3: Profile velocity mode (pv)
6: Homing mode (hm)
10: Cyclic torque mode (cst)

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6061_h$ | Modes of Operation Display | Variable | SIGNED8 |

*Table 107: Object Description ($6061_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Modes of Operation Display | no | 0 | 10 | 0 | | R |

*Table 108: Entry Description ($6061_h$)*

### 5.1.8 Object $608F_h$: Position Encoder Resolution

This object defines the resolution of the encoder. The position encoder resolution is calculated by the following formula: position encoder resolution = encoder increments / motor revolutions.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $608F_h$ | Position Encoder Resolution | Array | UNSIGNED32 |

*Table 109: Object Description ($608F_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | Encoder increments | no | 0 | 65535 | 4000 | | RW |
| 2 | Motor revolutions | no | 1 | 1 | 1 | | R |

*Table 110: Entry Description ($608F_h$)*

### 5.1.9 Object $6099_h$: Homing Speeds

This object indicates the configured speeds used during fast and slow homing procedure. In most homing modes, the home switch is searched with the fast speed first. When the home switch has been found, the motor will be decelerated to the slow speed (using the homing acceleration, object 609Ah) to search for the exact switch point. When the switch point has been found the motor will be stopped at that point.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6099_h$ | Homing Speeds | Array | UNSIGNED32 |

*Table 111: Object Description ($6099_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | Fast Homing Speed | no | 0 | 4294967295 | 0 | | RW |
| 2 | Slow Homing Speed | no | 0 | 4294967295 | 0 | | RW |

*Table 112: Entry Description ($6099_h$)*

### 5.1.10 Object $60FD_h$: Digital Inputs

This object contains the states of the digital inputs of the module. Starting from bit 0, every bit reflects the state of one digital input. The number of valid bits depends on the number of digital inputs of the module.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $60FD_h$ | Digital Inputs | Variable | UNSIGNED32 |

*Table 113: Object Description ($60FD_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Limit Switches | no | 0 | 3 | 0 | | R |

*Table 114: Entry Description ($60FD_h$)*

### 5.1.11 Object $6502_h$: Supported Drive Modes

This object provides information on the supported drive modes (0: not supported, 1: supported). This object is organized bit-wise. The bits have the following meaning:

Bit 0: profile position mode
Bit 1: velocity mode
Bit 2: profile velocity mode
Bit 3: profile torque mode
Bit 4: reserved
Bit 5: homing mode
Bit 6: interpolated position mode



Bit 7: cyclic synchronous position mode
Bit 8: cyclic synchronous velocity mode
Bit 9: cyclic synchronous torque mode
Bit 10-15: reserved
Bit 16-31: manufacturer-specific

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6502_h$ | Supported Drive Modes | Variable | UNSIGNED32 |

*Table 115: Object Description ($6502_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Supported Drive Modes | no | 0 | 4294967295 | 0 | | R |

*Table 116: Entry Description ($6502_h$)*

# 6 Profile Position Mode

A target position is applied to the trajectory generator. It is generating a position demand value for the position control loop described in the position control function.

Please refer to object $6060_h$ (section 5.1.6) for information about how to choose an operation mode. Object $6061_h$ (section 5.1.7) shows the operation mode that is set.

## 6.1 Detailed Object Specifications

The following text offers detailed object specifications. For a better understanding, it is necessary to see how the state machine works.

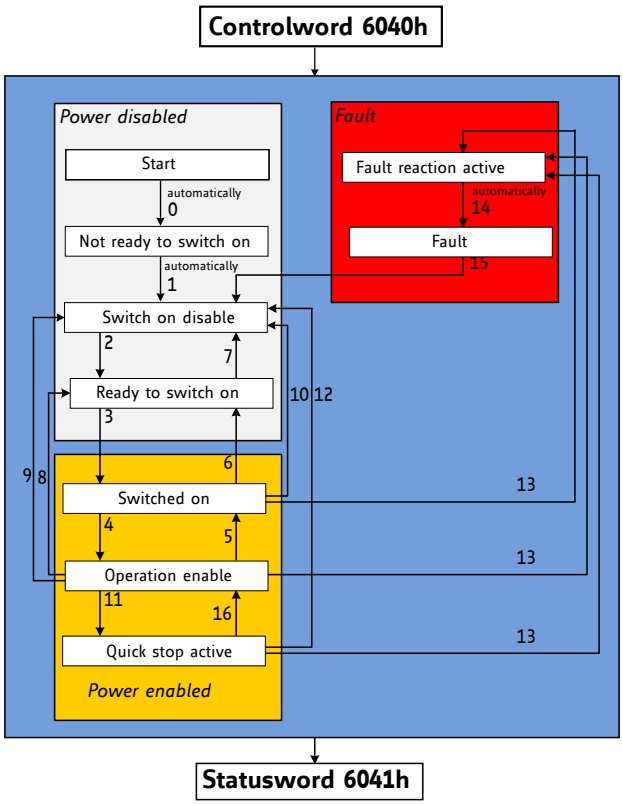



*Figure 4: DS402 Finite State Machine*

Notes on state transitions:

- Commands directing a change in state are processed completely and the new state achieved before additional state change commands are processed.
- Transitions 0 and 1 occur automatically at drive power-on or reset. Transition 14 occurs automatically, too. All other state changes must be directed by the host.
- Drive function disabled indicates that no current is being supplied to the motor.
- Drive function enabled indicates that current is available for the motor and profile position and profile velocity reference values may be processed.

## 6.2 Detailed Object Specifications

### 6.2.1 Object $6040_h$: Control Word

This object indicates the received command controlling the power drive system finite state automaton (PDS FSA). The CiA-402 state machine can be controlled using this object. Please refer to figure 4 for detailed information.

| Structure of the Control Word | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 15 11 | 10 | 9 | 8 | 7 | 6 4 | 3 | 2 | 1 | 0 | |
| nu | r | oms | h | fr | oms | eo | qs | ev | so | |
| MSB | | | | | | | | | | LSB |

Legend: nu=not used; r=reserved; oms=operation mode specific; h=halt; fr=fault reset; eo=enable operation; qs=quick stop; ev=enable voltage; so=switch on.

*Table 117: Structure of the Control Word in pp Mode*

| Operation Mode specific Bits in pp Mode | | |
|---|---|---|
| Bit | Name | Definition |
| 4 | New set point | 0-to-1: the next positioning will be started. |
| 5 | Change immediately | Not supported. |
| 6 | Absolute / relative | 0: New position is absolute.<br>1: New position is relative. |
| 9 | Change set point | Not supported. |

*Table 118: Operation Mode specific Bits in pp Mode*

| Command Coding | | | | | | |
|---|---|---|---|---|---|---|
| Command | Bits of Control Word | | | | | Transitions |
| | Bit 7 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| Shutdown | 0 | x | 1 | 1 | 0 | 2,6,8 |
| Switch on | 0 | 0 | 1 | 1 | 1 | 3 |
| Switch on & enable operation | 0 | 1 | 1 | 1 | 1 | 3, 4 |
| Disable voltage | 0 | x | x | 0 | x | 7,9,10,12 |
| Quick stop | 0 | x | 0 | 1 | x | 7,10,11 |
| Disable operation | 0 | 0 | 1 | 1 | 1 | 5 |
| Enable operation | 0 | 1 | 1 | 1 | 1 | 4, 16 |
| Fault reset | 0-to-1 | x | x | x | x | 15 |

*Table 119: Command Coding*

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6040_h$ | Controlword | Variable | UNSIGNED16 |

*Table 120: Object Description ($6040_h$ in pp Mode)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | see CiA402-3 | See command coding above. | |

*Table 121: Entry Description ($6040_h$ in pp Mode)*

### 6.2.2 Object $6041_h$: Status Word

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. Please refer to figure 4 for detailed information. The object is structured as defined below.
For more information about the coding please refer to the CANopen Drives and motion control device profile, part 2.

| Structure of the Status Word | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| dir | mot | oms | | ila | tr | rm | ms | w | sod | qs | ve | f | oe | so | rtso |
| MSB | | | | | | | | | | | | | | | LSB |

Legend: nu=not used; r=reserved; oms=operation mode specific; h=halt; fr=fault reset; oe=operation enable; qs=quick stop; ve=voltage enable; so=switch on.

*Table 122: Structure of the Staus Word in pp Mode*

| Trinamic Specific Bits | | |
|---|---|---|
| Bit | Name | Definition |
| 14 | Motor activity | 0: Motor stands still.<br>1: Motor rotates. |
| 15 | Direction of rotation | This bit shows the direction of rotation. |

*Table 123: Trinamic Specific Bits*

| Operation Mode specific Bits in pp Mode | | |
|---|---|---|
| Bit | Name | Definition |
| 10 | Target reached | Set when the motor is within the position window. |
| 12 | Set point acknowledged | 0: Set point processed.<br>1: Set point still in process. |
| 13 | Following error | Not supported. |

*Table 124: Operation Mode specific Bits in pp Mode*



| State Coding | |
|---|---|
| Status word | FSA state |
| xxxx xxxx x0xx $0000_h$ | Not ready to switch on |
| xxxx xxxx x1xx $0000_h$ | Switch on disabled |
| xxxx xxxx x01x $0001_h$ | Ready to switch on |
| xxxx xxxx x01x $0011_h$ | Switched on |
| xxxx xxxx x01x $0111_h$ | Operation enabled |
| xxxx xxxx x00x $0111_h$ | Quick stop active |
| xxxx xxxx x0xx $1111_h$ | Fault reaction active |
| xxxx xxxx x0xx $1000_h$ | Fault |

*Table 125: State Coding*

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6041_h$ | Controlword | Variable | UNSIGNED16 |

*Table 126: Object Description ($6041_h$ in pp Mode)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | see CiA402-3 | See state coding above. | |

*Table 127: Entry Description ($6041_h$ in pp Mode)*

### 6.2.3 Object $6062_h$: Position Demand Value

This object provides the demanded position value. The value is given in hall or encoder steps. Object 6062h indicates the actual position that the motor should have. It is not to be confused with objects 6063h and 6064h.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6062_h$ | Position Demand Value | Variable | SIGNED32 |

*Table 128: Object Description ($6062_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Position Demand Value | no | -2147483648 | 2147483647 | 0 | | R |

*Table 129: Entry Description ($6062_h$)*

### 6.2.4 Object $6063_h$: Position Actual Internal Value

This object provides the actual position value of the motor.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6063_h$ | Position Actual Internal Value | Variable | SIGNED32 |

*Table 130: Object Description ($6063_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Position Actual Internal Value | no | -2147483648 | 2147483647 | 0 | | R |

*Table 131: Entry Description ($6063_h$)*

### 6.2.5 Object $6064_h$: Position Actual Value

This object provides the actual value of the position measurement device. It always contains the same value as object 6063h.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6064_h$ | Position Actual Value | Variable | SIGNED32 |

*Table 132: Object Description ($6064_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Position Actual Value | no | -2147483648 | 2147483647 | 0 | | R |

*Table 133: Entry Description ($6064_h$)*



### 6.2.6 Object $6067_h$: Position Window

This object indicates the configured symmetrical range of accepted positions relative to the target position. If the actual value of the position encoder is within the position window, this target position is regarded as having been reached. The value is given in increments. If the value of the position window is FFFFFFFFh, the position window control is switched off. If this object is set to zero, the target reached event will be signaled when the demand position (6062h) has reached the target position (6064h). When the position window is set to a value greater than zero, the target reached event will be signaled when the actual encoder position value (6064h) is within (target_position - position_window) and (target_position + position_window).

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6067_h$ | Position Window | Variable | UNSIGNED32 |

*Table 134: Object Description ($6067_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Position Window | no | 0 | 4294967295 | 4294967295 | | RW |

*Table 135: Entry Description ($6067_h$)*

### 6.2.7 Object $606C_h$: Velocity Actual Value

This object shows the actual velocity value derived from the velocity sensor.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $606C_h$ | Velocity Actual Value | Variable | SIGNED32 |

*Table 136: Object Description ($606C_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Velocity Actual Value | no | -2147483648 | 2147483647 | 0 | [rpm] | R |

*Table 137: Entry Description ($606C_h$)*

### 6.2.8 Object $607A_h$: Target Position

The target position is the position that the drive should move to in profile position mode using the actual settings of motion control parameters (such as velocity, acceleration, deceleration, etc.). The value of this object is interpreted as absolute or relative depending on the abs/rel flag in the controlword.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $607A_h$ | Target Position | Variable | SIGNED32 |

*Table 138: Object Description ($607A_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Target Position | no | -2147483648 | 2147483647 | 0 | | RW |

*Table 139: Entry Description ($607A_h$)*

### 6.2.9 Object $607D_h$: Software Position Limit

This object indicates the configured maximal and minimal software position limits. These parameters define the absolute position limits for the position demand value and the position actual value. Every new target position is checked against these limits. The limit positions are always relative to the machine home position. Before being compared with the target position, they are corrected internally by the home offset as follows:

Corrected min position limit = min position limit - home offset
Corrected max position limit = max position limit - home offset

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $607D_h$ | Software Position Limit | Array | SIGNED32 |

*Table 140: Object Description ($607D_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | Min Position Limit | no | -2147483648 | 2147483647 | -2147483648 | | RW |
| 2 | Max Position Limit | no | -2147483648 | 2147483647 | 2147483647 | | RW |

*Table 141: Entry Description ($607D_h$)*

### 6.2.10 Object $6081_h$: Max Profile Velocity (pp)

This object indicates the configured velocity normally attained at the end of the acceleration ramp during a profiled motion and is valid for both directions of motion. The profile velocity is the maximum velocity used when driving to a new position.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6081_h$ | Max Profile Velocity (pp) | Variable | UNSIGNED32 |

*Table 142: Object Description ($6081_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Max Profile Velocity | no | 0 | 200000 | 0 | [rpm] | RW |

*Table 143: Entry Description ($6081_h$)*

### 6.2.11 Object $6082_h$: End Velocity

This object indicates the configured velocity normally attained at the end of the deceleration ramp during a profiled motion and is valid for both directions of motion. The end velocity is the velocity used when reaching the new position.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6082_h$ | End Velocity | Variable | SIGNED32 |

*Table 144: Object Description ($6082_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | End Velocity | no | -200000 | 200000 | 0 | [rpm] | RW |

*Table 145: Entry Description ($6082_h$)*

### 6.2.12 Object $6083_h$: Profile Acceleration

This object indicates the configured acceleration. Object 6083h sets the maximum acceleration to be used in profile positioning mode, and profile velocity mode.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6083_h$ | Profile Acceleration | Variable | UNSIGNED32 |

*Table 146: Object Description ($6083_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Profile Acceleration | no | 0 | 100000 | 2000 | [rpm/s] | RW |

*Table 147: Entry Description ($6083_h$)*

### 6.2.13 Object $6084_h$: Profile Deceleration

This object indicates the configured deceleration.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6084_h$ | Profile Deceleration | Variable | UNSIGNED32 |

*Table 148: Object Description ($6084_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Profile Deceleration | no | 0 | 100000 | 2000 | [rpm/s] | RW |

*Table 149: Entry Description ($6084_h$)*

### 6.2.14 Object $6085_h$: Quick Stop Deceleration

This object indicates the configured deceleration used to stop the motor when the quick stop function is activated and the quick stop code object 605Ah is set to 2 (or 6).

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6085_h$ | Quick Stop Deceleration | Variable | UNSIGNED32 |

*Table 150: Object Description ($6085_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Quick Stop Deceleration | no | 0 | 100000 | 2000 | [rpm/s] | RW |

*Table 151: Entry Description ($6085_h$)*



## 6.3 How to move a Motor in pp Mode

Here is a little example that shows how to get a motor running in pp mode. In this little example we assume that the module has been reset (and then switched to pre-operational or operational) by NMT commands before. Please note that the values are decimal.

- If you do not have any limit switches connected, first disable the limit switch inputs by writing 3 to object $2005_h$.
- Select pp mode by writing 1 to object $6060_h$.
- Write 6 to object $6040_h$ to switch to READY_TO_SWITCH_ON state.
- Write 7 to object $6040_h$ to switch to SWITCHED_ON state.
- Write 15 to object $6040_h$ to switch to OPERATION_ENABLED state.
- Write the desired target position (e.g. 500000) to object $607A_h$.
- Mark the new target position as active by writing 31 to object $6040_h$. The motor starts moving now.
- Reset the activation by writing 15 to object $6040_h$ (this can be done while the motor is still moving).

# 7 Profile Velocity Mode

The profile velocity mode is used to control the velocity of the drive without a special regard of the position. It contains limit functions and trajectory generation.

The profile velocity mode covers the following sub-functions:

- Demand value input via trajectory generator.
- Monitoring of the profile velocity using a window-function.
- Monitoring of velocity actual value using a threshold.

The operation of the reference value generator and its input parameters include:

- Profile velocity
- Profile acceleration
- Motion profile type

## 7.1 Detailed Object Specifications

### 7.1.1 Object $6040_h$: Control Word

This object indicates the received command controlling the power drive system finite state automaton (PDS FSA). The CiA-402 state machine can be controlled using this object. Please refer to figure 4 for detailed information.

In pv mode the control word does not contain any operation mode specific bits.

| Structure of the Control Word | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 11 | 10 | 9 | 8 | 7 | 6 | 4 | 3 | 2 | 1 | 0 |
| nu | | r | r | h | fr | r | | eo | qs | ev | so |
| MSB | | | | | | | | | | | LSB |

Legend: nu=not used; r=reserved; h=halt; fr=fault reset; eo=enable operation; qs=quick stop; ev=enable voltage; so=switch on.

*Table 152: Structure of the Control Word in pv Mode*

| Command Coding | | | | | | |
|---|---|---|---|---|---|---|
| Command | Bits of Control Word | | | | | Transitions |
| | Bit 7 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| Shutdown | 0 | x | 1 | 1 | 0 | 2,6,8 |
| Switch on | 0 | 0 | 1 | 1 | 1 | 3 |
| Switch on & enable operation | 0 | 1 | 1 | 1 | 1 | 3, 4 |
| Disable voltage | 0 | x | x | 0 | x | 7,9,10,12 |
| Quick stop | 0 | x | 0 | 1 | x | 7,10,11 |
| Disable operation | 0 | 0 | 1 | 1 | 1 | 5 |
| Enable operation | 0 | 1 | 1 | 1 | 1 | 4, 16 |
| Fault reset | 0-to-1 | x | x | x | x | 15 |

*Table 153: Command Coding*

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6040_h$ | Controlword | Variable | UNSIGNED16 |

*Table 154: Object Description ($6040_h$ in pv Mode)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | see CiA402-3 | See command coding above. | |

*Table 155: Entry Description ($6040_h$ in pv Mode)*

### 7.1.2 Object $6041_h$: Status Word

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. Please refer to figure 4 for detailed information. The object is structured as defined below.
For more information about the coding please refer to the CANopen Drives and motion control device profile, part 2.

| Structure of the Status Word | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| dir | mot | oms | | ila | tr | rm | ms | w | sod | qs | ve | f | oe | so | rtso |
| MSB | | | | | | | | | | | | | | | LSB |

Legend: nu=not used; r=reserved; oms=operation mode specific; h=halt; fr=fault reset; oe=operation enable; qs=quick stop; ve=voltage enable; so=switch on.

*Table 156: Structure of the Status Word in pv Mode*

| Trinamic Specific Bits | | |
|---|---|---|
| Bit | Name | Definition |
| 14 | Motor activity | 0: Motor stands still.<br>1: Motor rotates. |
| 15 | Direction of rotation | This bit shows the direction of rotation. |

*Table 157: Trinamic Specific Bits*

| Operation Mode specific Bits in pv Mode | | |
|---|---|---|
| Bit | Name | Definition |
| 10 | Target reached | Indicates that the target speed has been reached. |
| 12 | Speed | Not supported. |
| 13 | Max. slippage error | Not supported. |

*Table 158: Operation Mode specific Bits in pv Mode*

| State Coding | |
|---|---|
| Status word | FSA state |
| xxxx xxxx x0xx $0000_h$ | Not ready to switch on |
| xxxx xxxx x1xx $0000_h$ | Switch on disabled |
| xxxx xxxx x01x $0001_h$ | Ready to switch on |
| xxxx xxxx x01x $0011_h$ | Switched on |
| xxxx xxxx x01x $0111_h$ | Operation enabled |
| xxxx xxxx x00x $0111_h$ | Quick stop active |
| xxxx xxxx x0xx $1111_h$ | Fault reaction active |
| xxxx xxxx x0xx $1000_h$ | Fault |

*Table 159: State Coding*

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6041_h$ | Controlword | Variable | UNSIGNED16 |

*Table 160: Object Description ($6041_h$ in pv Mode)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | see CiA402-3 | See state coding above | |

*Table 161: Entry Description ($6041_h$ in pv Mode)*

### 7.1.3 Object $6062_h$: Position Demand Value

This object provides the demanded position value. The value is given in hall or encoder steps. Object 6062h indicates the actual position that the motor should have. It is not to be confused with objects 6063h and 6064h.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6062_h$ | Position Demand Value | Variable | SIGNED32 |

*Table 162: Object Description ($6062_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Position Demand Value | no | -2147483648 | 2147483647 | 0 | | R |

*Table 163: Entry Description ($6062_h$)*

### 7.1.4 Object $6063_h$: Position Actual Internal Value

This object provides the actual position value of the motor.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6063_h$ | Position Actual Internal Value | Variable | SIGNED32 |

*Table 164: Object Description ($6063_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Position Actual Internal Value | no | -2147483648 | 2147483647 | 0 | | R |

*Table 165: Entry Description ($6063_h$)*

### 7.1.5 Object $6064_h$: Position Actual Value

This object provides the actual value of the position measurement device. It always contains the same value as object 6063h.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6064_h$ | Position Actual Value | Variable | SIGNED32 |

*Table 166: Object Description ($6064_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Position Actual Value | no | -2147483648 | 2147483647 | 0 | | R |

*Table 167: Entry Description ($6064_h$)*

### 7.1.6 Object $606C_h$: Velocity Actual Value

This object shows the actual velocity value derived from the velocity sensor.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $606C_h$ | Velocity Actual Value | Variable | SIGNED32 |

*Table 168: Object Description ($606C_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Velocity Actual Value | no | -2147483648 | 2147483647 | 0 | [rpm] | R |

*Table 169: Entry Description ($606C_h$)*

### 7.1.7 Object $607D_h$: Software Position Limit

This object indicates the configured maximal and minimal software position limits. These parameters define the absolute position limits for the position demand value and the position actual value. Every new target position is checked against these limits. The limit positions are always relative to the machine home position. Before being compared with the target position, they are corrected internally by the home offset as follows:

Corrected min position limit = min position limit - home offset
Corrected max position limit = max position limit - home offset



| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $607D_h$ | Software Position Limit | Array | SIGNED32 |

*Table 170: Object Description ($607D_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | Min Position Limit | no | -2147483648 | 2147483647 | -2147483648 | | RW |
| 2 | Max Position Limit | no | -2147483648 | 2147483647 | 2147483647 | | RW |

*Table 171: Entry Description ($607D_h$)*

### 7.1.8 Object $6083_h$: Profile Acceleration

This object indicates the configured acceleration. Object 6083h sets the maximum acceleration to be used in profile positioning mode, and profile velocity mode.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6083_h$ | Profile Acceleration | Variable | UNSIGNED32 |

*Table 172: Object Description ($6083_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Profile Acceleration | no | 0 | 100000 | 2000 | [rpm/s] | RW |

*Table 173: Entry Description ($6083_h$)*

### 7.1.9 Object $6085_h$: Quick Stop Deceleration

This object indicates the configured deceleration used to stop the motor when the quick stop function is activated and the quick stop code object 605Ah is set to 2 (or 6).

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6085_h$ | Quick Stop Deceleration | Variable | UNSIGNED32 |

*Table 174: Object Description ($6085_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Quick Stop Deceleration | no | 0 | 100000 | 2000 | [rpm/s] | RW |

*Table 175: Entry Description ($6085_h$)*

### 7.1.10 Object $60FF_h$: Target Velocity

This object indicates the configured target velocity and is used as input for the trajectory generator. Object 60FFh sets the target velocity when using profile velocity mode. The drive then accelerates or decelerates to that velocity using the acceleration and deceleration set by objects 6083h and 6084h.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $60FF_h$ | Target Velocity | Variable | SIGNED32 |

*Table 176: Object Description ($60FF_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Target Velocity | no | -200000 | 200000 | 0 | [rpm] | RW |

*Table 177: Entry Description ($60FF_h$)*

## 7.2 How to move a Motor in pv Mode

Here is a little example that shows how to get a motor running in pv mode. In this little example we assume that the module has been reset (and then switched to pre-operational or operational) by NMT commands before.

- Select pv mode by writing 3 to object $6060_h$ (Modes_of_Operation).
- Write 6 to object $6040_h$ (Controlword) to switch to READY_TO_SWITCH_ON state.
- Write 7 to object $6040_h$ to switch to SWITCHED_ON state.
- Write the desired target velocity (e.g. 2000) to object $60FF_h$ (Target_Velocity).
- Write 15 to object $6040_h$ to switch to OPERATION_ENABLED state. The motor now accelerates to the target velocity.
- Stop the motor by writing 0 to object $60FF_h$.

# 8 Homing mode

This chapter describes the method by which a drive seeks the home position (reference point). There are various methods of achieving this using limit switches at the ends of travel or a home switch in mid-travel. Some methods also use the index (zero) pulse train from an incremental encoder. The user may specify the speeds, acceleration and the method of homing.

There is no output data except for those bits in the statusword which return the status or result of the homing process and the demand to the position control loops.

There are four sources of the homing signal available: these are positive and negative limit switches, the home switch and the index pulse from an encoder.

Figure 5 shows the defined input objects as well as the output objects. The user can specify the speeds, acceleration and method of homing. The home offset object $607C_h$ allows displacing the zero in point the coordinate system for the home position.

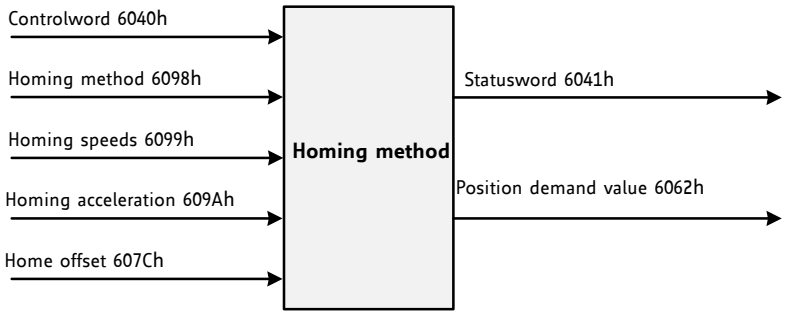



*Figure 5: Homing Mode Function*

Choosing a homing mode determines the following things:

- The homing signal (positive limit switch, negative limit switch, and home switch).
- The direction of actuation where appropriate.
- The position of the index pulse.

The home position and the zero position are offset by the home offset (see object $607C_h$, section 8.2.4).

There are four sources of homing signals available:

- Negative and positive limit switches.
- Home switch.
- Index pulse of an encoder.

For the operation of positioning drives, an exact knowledge of the absolute position is normally required. Since for cost reasons drives often do not have an absolute encoder, a homing operation is necessary.

## 8.1 Homing Methods

The TMCM-1633 supports a subset of different standard CANopen homing methods. The homing method that is to be used can be choosen via object $6098_h$ (section 8.2.5).

| Supported Homing Methods | |
|---|---|
| Method | Description |
| 0 | No homing (default value for object $6098_h$). |
| 17 | Search the left end switch. |
| 18 | Search the right end switch. |
| 35 | The actual position is used as home position. All position values (objects 6062h, 6063h, and 6064h) are set to zero, but the motor will not move. |

*Table 178: Supported CANopen Homing Methods*

When using homing methods that need end switch inputs or home switch inputs please take care of their configuration (object $2005_h$, section 4.1.1).

### 8.1.1 Homing Method 17 and 18: Homing without Index Pulse

These methods are similar to methods 1 to 5 except that the home position is not dependent on the index pulse but only dependent on the relevant home or limit switch transitions.

| Homing Methods 17…21 | |
|---|---|
| Method | Description |
| 17 | Search the left end switch. (Similar to method 1) |
| 18 | Search the right end switch. (Similar to method 2) |

*Table 179: Homing Methods 17 – 21*

### 8.1.2 Homing Method 35: Current Position as Home Position

In this method, the current position shall be taken to be the home position. This method does not require the drive device to be in operation enabled state.

## 8.2 Detailed Object Specifications

### 8.2.1 Object $6040_h$: Control Word

This object indicates the received command controlling the power drive system finite state automaton (PDS FSA). The CiA-402 state machine can be controlled using this object. Please refer to figure 4 for detailed information.

| Structure of the Control Word | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 15 11 | 10 | 9 | 8 | 7 | 6 4 | 3 | 2 | 1 | 0 | |
| nu | r | oms | h | fr | oms | eo | qs | ev | so | |
| MSB | | | | | | | | | | LSB |

Legend: nu=not used; r=reserved; oms=operation mode specific; h=halt; fr=fault reset; eo=enable operation; qs=quick stop; ev=enable voltage; so=switch on.

*Table 180: Structure of the Control Word in hm Mode*

| Operation Mode specific Bits in hm Mode | | |
|---|---|---|
| Bit | Name | Definition |
| 4 | Homing operation start | 1: start homing; 0: stop homing |
| 8 | Halt | Not supported. |

*Table 181: Operation Mode specific Bits in hm Mode*

| Command Coding | | | | | | |
|---|---|---|---|---|---|---|
| Command | Bits of Control Word | | | | | Transitions |
| | Bit 7 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| Shutdown | 0 | x | 1 | 1 | 0 | 2,6,8 |
| Switch on | 0 | 0 | 1 | 1 | 1 | 3 |
| Switch on & enable operation | 0 | 1 | 1 | 1 | 1 | 3, 4 |
| Disable voltage | 0 | x | x | 0 | x | 7,9,10,12 |
| Quick stop | 0 | x | 0 | 1 | x | 7,10,11 |
| Disable operation | 0 | 0 | 1 | 1 | 1 | 5 |
| Enable operation | 0 | 1 | 1 | 1 | 1 | 4, 16 |
| Fault reset | 0-to-1 | x | x | x | x | 15 |

*Table 182: Command Coding*



| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6040_h$ | Controlword | Variable | UNSIGNED16 |

*Table 183: Object Description ($6040_h$ in hm Mode)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | see CiA402-3 | See command coding above. | |

*Table 184: Entry Description ($6040_h$ in hm Mode)*

### 8.2.2 Object $6041_h$: Status Word

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. Please refer to figure 4 for detailed information. The object is structured as defined below.
For more information about the coding please refer to the CANopen Drives and motion control device profile, part 2.

| Structure of the Status Word | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| dir | mot | oms | | ila | tr | rm | ms | w | sod | qs | ve | f | oe | so | rtso |
| MSB | | | | | | | | | | | | | | | LSB |

Legend: nu=not used; r=reserved; oms=operation mode specific; h=halt; fr=fault reset; oe=operation enable; qs=quick stop; ve=voltage enable; so=switch on.

*Table 185: Structure of the Status Word in hm Mode*

| Trinamic Specific Bits | | |
|---|---|---|
| Bit | Name | Definition |
| 14 | Motor activity | 0: Motor stands still.<br>1: Motor rotates. |
| 15 | Direction of rotation | This bit shows the direction of rotation. |

*Table 186: Trinamic Specific Bits*

| Operation Mode specific Bits in hm Mode | | |
|---|---|---|
| Bit | Name | Definition |
| 10 | Target reached | Set when the zero position has been found or homing has been stopped by setting controlword bit 4 to zero. |
| 12 | Home attained | Set when zero position has been found. |
| 13 | Homing error | Not supported. |

*Table 187: Operation Mode specific Bits in hm Mode*

| State Coding | |
|---|---|
| Status word | FSA state |
| xxxx xxxx x0xx $0000_h$ | Not ready to switch on |
| xxxx xxxx x1xx $0000_h$ | Switch on disabled |
| xxxx xxxx x01x $0001_h$ | Ready to switch on |
| xxxx xxxx x01x $0011_h$ | Switched on |
| xxxx xxxx x01x $0111_h$ | Operation enabled |
| xxxx xxxx x00x $0111_h$ | Quick stop active |
| xxxx xxxx x0xx $1111_h$ | Fault reaction active |
| xxxx xxxx x0xx $1000_h$ | Fault |

*Table 188: State Coding*

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6041_h$ | Controlword | Variable | UNSIGNED16 |

*Table 189: Object Description ($6041_h$ in hm Mode)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | see CiA402-3 | See state coding above. | |

*Table 190: Entry Description ($6041_h$ in hm Mode)*

### 8.2.3 Object $606C_h$: Velocity Actual Value

This object shows the actual velocity value derived from the velocity sensor.



| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $606C_h$ | Velocity Actual Value | Variable | SIGNED32 |

*Table 191: Object Description ($606C_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Velocity Actual Value | no | -2147483648 | 2147483647 | 0 | [rpm] | R |

*Table 192: Entry Description ($606C_h$)*

### 8.2.4 Object $607C_h$: Home Offset

This object indicates the configured difference between the zero position for the application and the machine home position/home switch (found during homing). While homing, the machine home position is found and once the homing is completed, the zero position is offset from the home position by adding the home offset to the home position. The effect of setting the home position to a non-zero value depends on the selected homing method. Negative values indicate the opposite direction.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $607C_h$ | Home Offset | Variable | SIGNED32 |

*Table 193: Object Description ($607C_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Home Offset | no | -2147483648 | 2147483647 | 0 | | RW |

*Table 194: Entry Description ($607C_h$)*

### 8.2.5 Object $6098_h$: Homing Method

The actual homing method.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6098_h$ | Homing Method | Variable | SIGNED8 |

*Table 195: Object Description ($6098_h$)*



| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Homing Method | no | 0 | 35 | 0 | | RW |

*Table 196: Entry Description ($6098_h$)*

### 8.2.6 Object $6099_h$: Homing Speeds

This object indicates the configured speeds used during fast and slow homing procedure. In most homing modes, the home switch is searched with the fast speed first. When the home switch has been found, the motor will be decelerated to the slow speed (using the homing acceleration, object 609Ah) to search for the exact switch point. When the switch point has been found the motor will be stopped at that point.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6099_h$ | Homing Speeds | Array | UNSIGNED32 |

*Table 197: Object Description ($6099_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 1 | Fast Homing Speed | no | 0 | 4294967295 | 0 | | RW |
| 2 | Slow Homing Speed | no | 0 | 4294967295 | 0 | | RW |

*Table 198: Entry Description ($6099_h$)*

### 8.2.7 Object $609A_h$: Homing Acceleration

This object indicates the configured acceleration and deceleration to be used during homing operation.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $609A_h$ | Homing Acceleration | Variable | UNSIGNED32 |

*Table 199: Object Description ($609A_h$)*



| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Homing Acceleration | no | 0 | 100000 | 2000 | [rpm/s] | RW |

*Table 200: Entry Description ($609A_h$)*

### 8.2.8 Object $2100_h$: Home Offset Display

This object shows the home offset. The value is given in encoder or hall increments.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $2100_h$ | Home Offset Display | Variable | SIGNED32 |

*Table 201: Object Description ($2100_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Home Offset Display | no | -2147483648 | 2147483647 | 0 | | R |

*Table 202: Entry Description ($2100_h$)*



## 8.3 How to start a Homing in hm Mode

Here is a little example that shows how to home the motor in hm mode. In this little example we assume that the module has been reset (and then switched to pre-operational or operational) by NMT commands before. The home switch must be connected to the home switch input. It can be operated manually.

- Select hm mode by writing 6 to object $6060_h$.
- Write 6 to object $6040_h$ to switch to READY_TO_SWITCH_ON state.
- Write 7 to object $6040_h$ to switch to SWITCHED_ON state.
- Write 15 to object $6040_h$ to switch to OPERATION_ENABLED state.
- Select homing method 17 (or 18) by writing 17 (or 18) to object $6098_h$.
- Set the homing speeds by writing e.g. 500 to object $6099_h$ sub index 1 and e.g. 200 to object $6099_h$ sub index 2.
- Write 31 to object $6040_h$ to start the homing process.
- Press and release the home switch.
- When homing has finished, write 15 to object $6040_h$ again.

# 9 Cyclic synchonous Torque Mode

The cyclic synchronous torque mode is used to directly control the torque of the motor, without the need for position or velocity control. It contains limit functions, but not a trajectory generator.

The cyclic synchronous torque mode covers the following sub-functions:

- Demand value input directly via an object.
- Monitoring of the torque.
- Limiting the position using the software limits or the hardware limit switches.

## 9.1 Detailed Object Specifications

### 9.1.1 Object $6040_h$: Control Word

This object indicates the received command controlling the power drive system finite state automaton (PDS FSA). The CiA-402 state machine can be controlled using this object. Please refer to figure 4 for detailed information. The cyclic synchronous torque mode does not use any mode specific bits of the control word.

| Structure of the Control Word | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 9 | 8 | 7 | 6 | 4 | 3 | 2 | 1 | 0 | |
| nu | | h | fr | nu | | eo | qs | ev | so | |
| MSB | | | | | | | | | | LSB |

Legend: nu=not used; h=halt; fr=fault reset; eo=enable operation; qs=quick stop; ev=enable voltage; so=switch on.

*Table 203: Structure of the Control Word in cst Mode*

| Command Coding | | | | | | |
|---|---|---|---|---|---|---|
| Command | Bits of Control Word | | | | | Transitions |
| | Bit 7 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| Shutdown | 0 | x | 1 | 1 | 0 | 2,6,8 |
| Switch on | 0 | 0 | 1 | 1 | 1 | 3 |
| Switch on & enable operation | 0 | 1 | 1 | 1 | 1 | 3, 4 |
| Disable voltage | 0 | x | x | 0 | x | 7,9,10,12 |
| Quick stop | 0 | x | 0 | 1 | x | 7,10,11 |
| Disable operation | 0 | 0 | 1 | 1 | 1 | 5 |
| Enable operation | 0 | 1 | 1 | 1 | 1 | 4, 16 |
| Fault reset | 0-to-1 | x | x | x | x | 15 |

*Table 204: Command Coding*

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6040_h$ | Controlword | Variable | UNSIGNED16 |

*Table 205: Object Description ($6040_h$ in cst Mode)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | see CiA402-3 | See command coding above. | |

*Table 206: Entry Description ($6040_h$ in cst Mode)*

### 9.1.2 Object $6041_h$: Status Word

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. Please refer to figure 4 for detailed information. The object is structured as defined below.

For more information about the coding please refer to the CANopen Drives and motion control device profile, part 2.

| Structure of the Status Word | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| dir | mot | oms | | ila | r | rm | ms | w | sod | qs | ve | f | oe | so | rtso |
| MSB | | | | | | | | | | | | | | | LSB |

Legend: nu=not used; r=reserved; oms=operation mode specific; h=halt; fr=fault reset; oe=operation enable; qs=quick stop; ve=voltage enable; so=switch on.

*Table 207: Structure of the Status Word in cst Mode*

| Trinamic Specific Bits | | |
|---|---|---|
| Bit | Name | Definition |
| 14 | Motor activity | 0: Motor stands still.<br>1: Motor rotates. |
| 15 | Direction of rotation | This bit shows the direction of rotation. |

*Table 208: Trinamic Specific Bits*

| Operation Mode specific Bits in cst Mode | | |
|---|---|---|
| Bit | Name | Definition |
| 10 | Reserved | Not used. |
| 12 | Target torque ignored | 0: Target torque ignored.<br>1: Target torque used as input to control loop. |
| 13 | Reserved | Not used. |

*Table 209: Operation Mode specific Bits in cst Mode*

| State Coding | |
|---|---|
| Status word | FSA state |
| xxxx xxxx x0xx $0000_h$ | Not ready to switch on |
| xxxx xxxx x1xx $0000_h$ | Switch on disabled |
| xxxx xxxx x01x $0001_h$ | Ready to switch on |
| xxxx xxxx x01x $0011_h$ | Switched on |
| xxxx xxxx x01x $0111_h$ | Operation enabled |
| xxxx xxxx x00x $0111_h$ | Quick stop active |
| xxxx xxxx x0xx $1111_h$ | Fault reaction active |
| xxxx xxxx x0xx $1000_h$ | Fault |

*Table 210: State Coding*

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6041_h$ | Controlword | Variable | UNSIGNED16 |

*Table 211: Object Description ($6041_h$ in cst Mode)*

| Entry Description | | | | |
|---|---|---|---|---|
| Sub-index | Access | PDO Mapping | Value Range | Default Value |
| 0 | rw | see CiA402-3 | See state coding above | |

*Table 212: Entry Description ($6041_h$ in cst Mode)*

### 9.1.3 Object $6071_h$: Target Torque

This object gives the target motor current.



| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6071_h$ | Target Torque | Variable | SIGNED32 |

*Table 213: Object Description ($6071_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Target Torque | no | -15000 | 15000 | 0 | [mA] | RW |

*Table 214: Entry Description ($6071_h$)*

### 9.1.4 Object $6077_h$: Torque Actual Value

The actual motor current.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $6077_h$ | Torque Actual Value | Variable | SIGNED32 |

*Table 215: Object Description ($6077_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Torque Actual Value | no | -2147483648 | 2147483647 | 0 | | R |

*Table 216: Entry Description ($6077_h$)*

### 9.1.5 Object $60B2_h$: Torque offset

The actual set torque offset.

| Object Description | | | |
|---|---|---|---|
| Index | Name | Object Type | Data Type |
| $60B2_h$ | Torque offset | Variable | SIGNED32 |

*Table 217: Object Description ($60B2_h$)*

| Entry Description | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sub-index | Name | PDO Mapping | Min | Max | Default | Unit | Access |
| 0 | Torque offset | no | -2147483648 | 2147483647 | 0 | | RW |

*Table 218: Entry Description ($60B2_h$)*

## 9.2 How to move a Motor in cst Mode

Here is a little example that shows how to get a motor running in cst mode. In this little example we assume that the module has been reset (and then switched to start) by NMT commands before.

- If you do not have any limit switches connected, first disable the limit switch inputs by writing 3 to object $2005_h$.
- Select cst mode by writing 10 to object $6060_h$.
- Write 6 to object $6040_h$ to switch to READY_TO_SWITCH_ON state.
- Write 7 to object $6040_h$ to switch to SWITCHED_ON state.
- Write 15 to object $6040_h$ to switch to OPERATION_ENABLED state.
- Write the desired torque (e.g. 1000) to object $6071_h$ to start the motor.
- To stop the motor, write 0 to object $6071_h$.

# 10 Emergency Messages (EMCY)

The module sends an emergency message if an error occurs. The message contains information about the error type. The module can map internal errors and object $1001_h$ (error register) is part of every emergency object.

| Emergency Messages (EMCY) of the TMCM-1633 | | | | | | |
|---|---|---|---|---|---|---|
| Error code | Additional byte | | | | | Description |
| | 1 | 2 | 3 | 4 | 5 | |
| $0000_h$ | 0 | 0 | 0 | 0 | 0 | **Fault reset**<br>The fault reset command has been executed. |
| $1000_h$ | 1 | 0 | 0 | 0 | 0 | **Generic error: open load bridge A**<br>The motor driver indicates open load on bridge A. It is possible that the motor cable is broken or that there is an error in the power amplifier itself. |
| $1000_h$ | 2 | 0 | 0 | 0 | 0 | **Generic error: open load bridge B**<br>The motor driver indicates open load on bridge B. It is possible that the motor cable is broken or that there is an error in the power amplifier itself. |
| $2310_h$ | 0 | 0 | 0 | 0 | 0 | **Overcurrent high side**<br>The motor driver indicates an overcurrent on the high side. This can be caused by a short circuit in the driver stage. |
| $2311_h$ | 0 | 0 | 0 | 0 | 0 | **Overcurrent bridge B**<br>The motor driver indicates that there is overcurrent on bridge B. This can be caused by a short circuit in the motor itself or in the motor driver stage. |
| $2312_h$ | 0 | 0 | 0 | 0 | 0 | **Overcurrent bridge A**<br>The motor driver indicates that there is overcurrent on bridge A. This can be caused by a short circuit in the motor itself or in the motor driver stage. |
| $3230_h$ | 0 | 0 | 0 | 0 | 0 | **stallGuard2 error**<br>The actual load value exceeds the stallGuard2 limit. |
| $4310_h$ | 1 | 0 | 0 | 0 | 0 | **Overtemperature pre-warning**<br>The temperature in the motor driver exceeds the pre-warning limit. |
| $4310_h$ | 2 | 0 | 0 | 0 | 0 | **Overtemperature error**<br>The motor driver has been switched off because the temperature limit has been exceeded. |
| $5441_h$ | 0 | 255 | 0 | 0 | 0 | **Shutdown switch active**<br>The enable signal is missing (due to the shutdown switch) and the motor driver has been switched off. |
| $6320_h$ | 0 | 255 | 0 | 0 | 0 | **Parameter error**<br>The data in the received PDO is either wrong or cannot be accepted due to the internal state of the drive. |

| Error code | Additional byte | | | | | Description |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| $8110_h$ | 1 | 255 | 0 | 0 | 0 | **CAN controller overflow**<br>The receive message buffer of the CAN controller hardware is full and some CAN messages are lost. |
| $8110_h$ | 2 | 255 | 0 | 0 | 0 | **CAN Tx buffer overflow**<br>The software CAN transmit buffer is full and thus some CAN messages are lost. |
| $8110_h$ | 3 | 255 | 0 | 0 | 0 | **CAN Rx buffer overflow**<br>The software CAN receive buffer is full and so some CAN messages are lost. |
| $8120_h$ | 0 | 255 | 0 | 0 | 0 | **CAN error passive**<br>The CAN controller has detected communication errors and has entered the CAN Error passive state. |
| $8140_h$ | 0 | 255 | 0 | 0 | 0 | **CAN controller recovered from bus-off state**<br>The CAN controller has detected too many errors and has changed into the bus-off state. The drive has been stopped and disabled. This message is sent after the CAN controller has recovered from bus-off state and is bus-on again. |
| $8611_h$ | 0 | 0 | 0 | 0 | 0 | **Following error**<br>The deviation between motor position counter and encoder position counter has exceeded the following error window. |
| $ff00_h$ | 0 | 0 | 0 | 0 | 0 | **Undervoltage**<br>The supply voltage is too low to drive a motor. |
| $ff01_h$ | 1 | 0 | 0 | 0 | 0 | **Positive software limit**<br>The actual position is outside the range defined by object $607d_h$. |
| $ff01_h$ | 2 | 0 | 0 | 0 | 0 | **Negative software limit**<br>The actual position is outside the range defined by object $607d_h$. |
| $ff01_h$ | 3 | 0 | 0 | 0 | 0 | **Positive limit switch**<br>The positive limit switch has been touched outside of the homing function. |
| $ff01_h$ | 4 | 0 | 0 | 0 | 0 | **Negative limit switch**<br>The negative limit switch has been touched outside of the homing function. |

*Table 219: Emergency Messages (EMCY)*

# 11 Figures Index





# 12 Tables Index





# 13 Supplemental Directives

## 13.1 Producer Information

## 13.2 Copyright



## 13.3 Trademark Designations and Symbols

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

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

## 13.4 Target User

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

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

## 13.5 Disclaimer: Life Support Systems

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

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

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

## 13.6 Disclaimer: Intended Use

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



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

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

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

## 13.7 Collateral Documents & Tools

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

# 14 Revision History

## 14.1 Firmware Revision

| Version | Date | Author | Description |
|---|---|---|---|
| 2.09 | 28.06.2017 | ED | First release. |

*Table 220: Firmware Revision*

## 14.2 Document Revision

| Version | Date | Author | Description |
|---|---|---|---|
| 1.00 | 28.06.2017 | ED | First release. |

*Table 221: Document Revision*