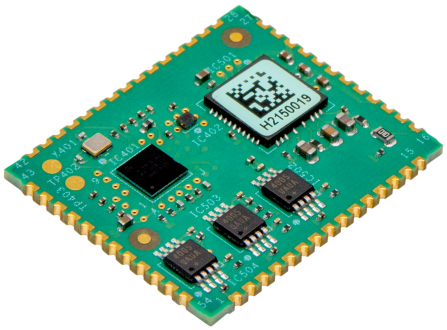
TMCM-1690 CANopen Firmware Manual

Firmware Version V1.03 | Rev 1: 04/25

The TMCM-1690 is a single axis FOC servo controller gate driver module for 3-phase BLDC and DC motors with up to 1.5A gate drive current and +60V (+48 V nominal) supply. It offers UART (RS232), CAN, and EtherCAT® interfaces with TMCL, CANopen, or CANopen-over-EtherCAT protocol support for communication. The TMCM-1690 supports incremental encoders, digital hall sensors, and absolute encoder as position feedback.



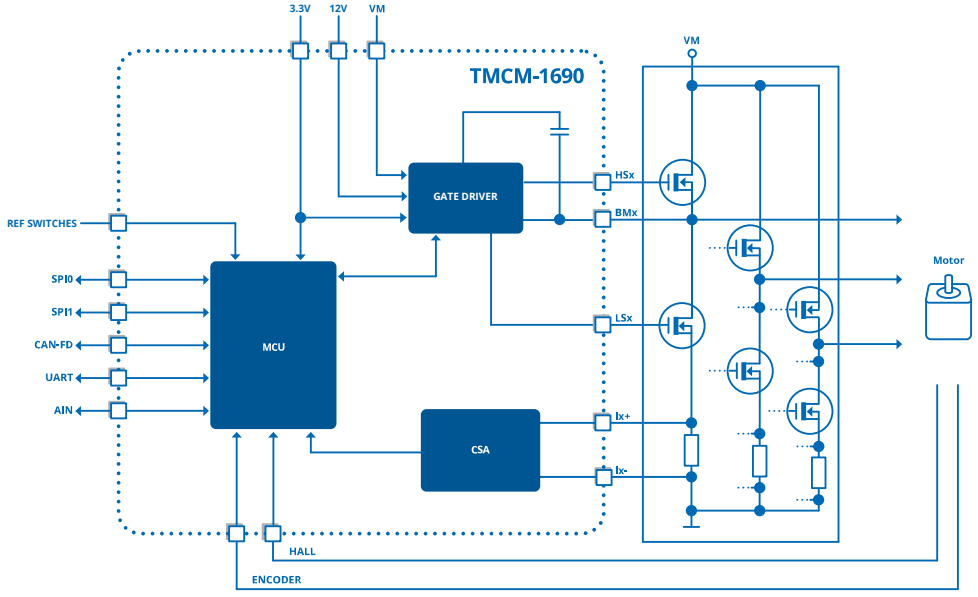
Features

- Supply voltage +10V to +60V DC
- FOC servo controller gate driver module for BLDC and DC motors
- 0.5A/1.0A/1.5A gate drive current
- Up to 120kHz PWM frequency
- Onboard current-sense amplifiers
- Supports UART (RS232), CAN, and EtherCAT™ interface
- Supports incremental encoders (ABN), digital HALL sensors, absolute (SPI/SSI) encoders
- Reference switch inputs
- Compact size 27mm x 22.5mm

Applications

- Robotics
- Laboratory Automation
- Manufacturing
- Factory Automation
- Servo Drives
- Motorized Tables and Chairs
- Industrial BLDC and DC Motor Drives

Simplified Block Diagram



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Read entire documentation.

Contents

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

1 Preface

This document specifies objects and modes of operation of the Trinamic TMCM-1690 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-1690 TMCL fimware again through the CAN interface or the UART (RS232) interface, with the help of the firmware update function of the TMCL-IDE (3.0 or higher).

1.1 General Features of this CANopen Implementation

Main Characteristics

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

SDO Communication

- One server
- Expedited transfer
- Segmented transfer
- No block transfer

PDO Communication

- Producer
- Consumer
- RPDOs
  - Axis 0: 1, 2, 3, 4
  - Transmission modes: asynchronous, synchronous.
  - Dynamic mapping with max. three 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. three 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 and RPDOs can be configured as synchronous PDOs)
- Emergency: Producer
- RTR: Not supported
- 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. The new firmware can be loaded into the module through the firmware update function of the TMCL-IDE, using the CAN interface or the RS232 interface of the module.

2 Motor Regulation

2.1 Structure of Motor Control Regulation Modes

The TMCM-1690 supports a current, velocity, and position PI regulation mode for motor control in different application areas. These regulation modes are shown in Figure 1. Individual modes are explained in the following subsections.

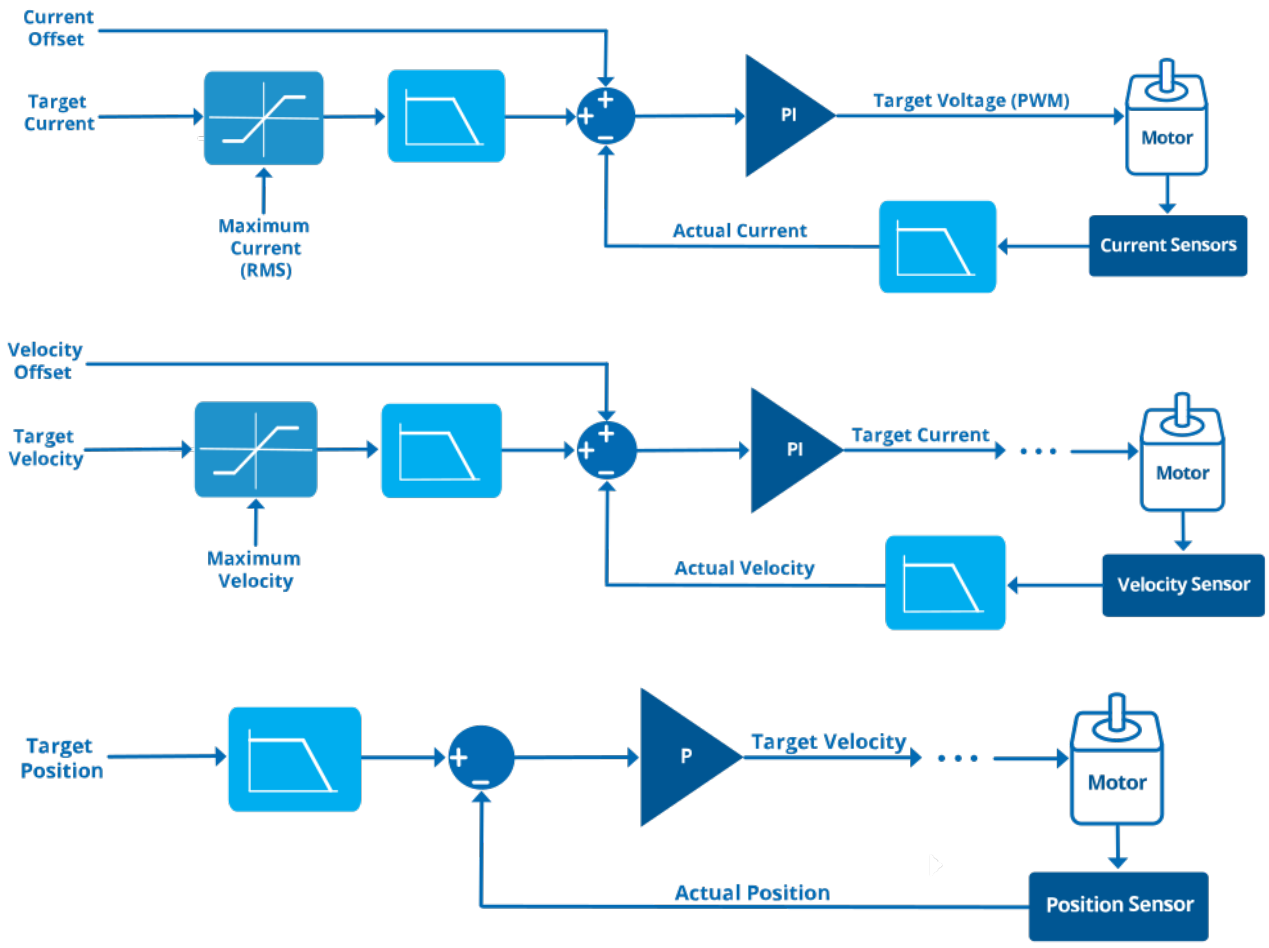


Figure 1: Motion Regulation

It can be seen that the target current signal is first limited by maximum current. There is an option for a filter (average or biquad) to this signal. This is followed by having an impact of P and I parameters to the signal and it is fed to the motor. There is a feedback coming from the actual current forming a closed loop. The velocity and position regulation are performed in the same fashion.

2.2 Current Regulation

The current regulation mode uses a FOC current and flux regulator to adjust a desired motor current. The current loop is running typically at 20kHz, and the PWM loop is variable and can be set using object 2003_h sub-object **PWM frequency** The PWM frequency ranges from 20kHz to 120kHz. The target current can be set by object 6071_h **(Target torque)**. The maximal target current is limited by object 2003_h **(Drive**

settings) sub-object **Maximum current**. The current regulation uses two basic parameters: the P and I parameters.

2.2.1 Structure of the Current Regulator

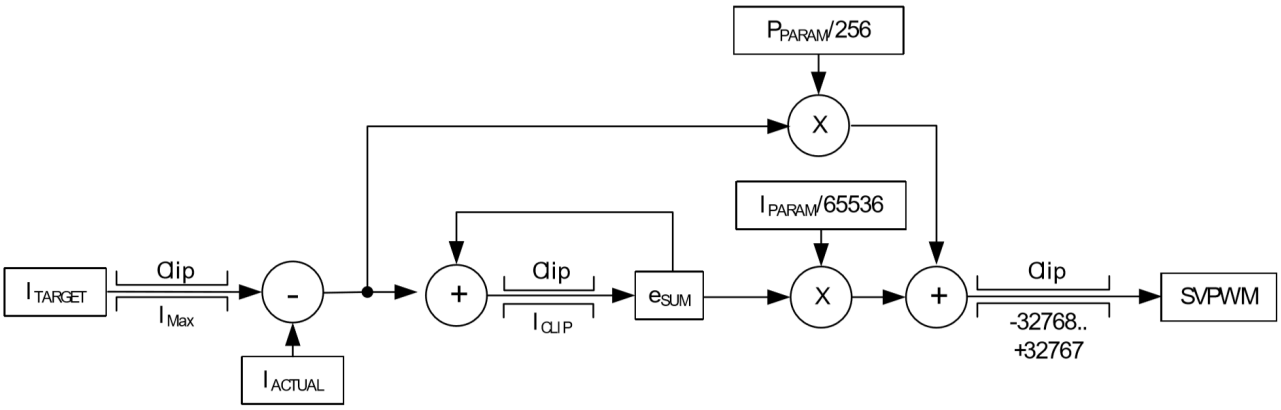


Figure 2: Current Regulation (See Parameter Descriptions in Table 2)

Current Regulation Parameters	
Parameter	Description
I_{ACTUAL}	Actual current (Object 6077_h)
I_{TARGET}	Target current (Object 6071_h)
I_{Max}	Max. target current (Object 2003_h, sub-object **Maximum current**)
e_{SUM}	Error sum for integral calculation
P_{PARAM}	Current P parameter (Object $201E_h$, sub-object **P**)
I_{PARAM}	Current I parameter (Object $201E_h$, sub-object **I**)

Table 2: Current Regulation Parameters

2.2.1.1 Parametrizing the Current Regulator Set

Parametrizing the current regulator set is preferably done with the PI tuning tool using auto tuning. See further details in the TMCL-IDE documentation under the section PI tuning. However, it can be done manually as well without the tool. To parameterize the current regulator set properly, do as follows:

1. Set the P parameter and the I parameter to zero.
2. Start the motor by using a low target flux (example, 1000mA).
3. Modify the current P parameter. Start from a low value and go to a higher value, until the actual current nearly reaches 70% of the desired target current.
4. Do the same with the current I parameter.

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

2.3 Velocity Regulation

Based on the current regulation, the motor velocity can be controlled by the velocity PI regulator. The velocity regulation runs in the velocity loop clocked at 2kHz

2.3.1 Structure of the Velocity Regulator

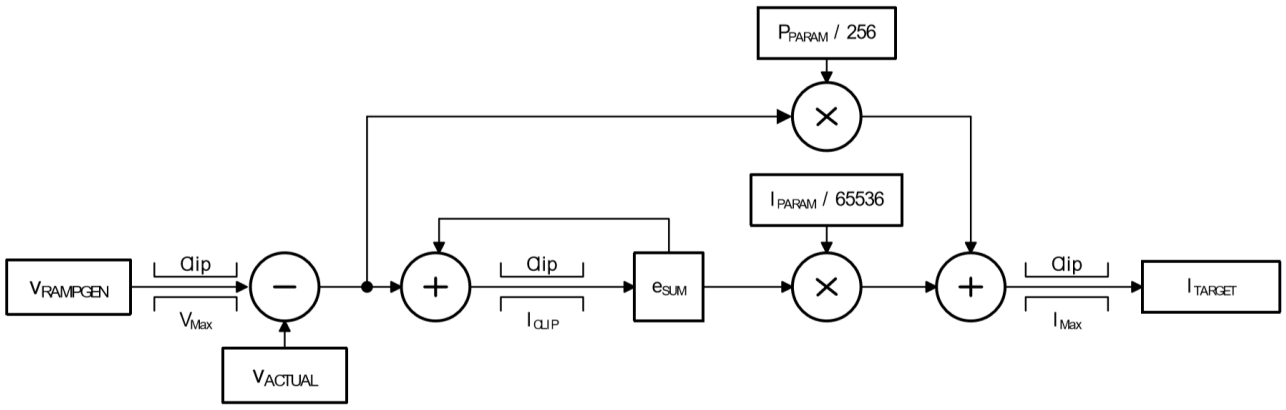


Figure 3: Velocity Regulation (See Parameter Descriptions in Table 3)

Velocity Regulation Parameters	
Parameter	Description
V_{ACTUAL}	Actual motor velocity (Object $606C_h$)
$V_{RAMPGEN}$	Target velocity of ramp generator (Object $60FF_h$)
V_{Max}	Max. target velocity (Object $607F_h$)
e_{SUM}	Error sum for integral calculation
P_{PARAM}	Velocity P parameter (Object 2023_h, sub-object **P**)
I_{PARAM}	Velocity I parameter (Object 2023_h, sub-object **I**)
I_{Max}	Max. target current (Object 2003_h, sub-object **Maximum current**)

Table 3: Velocity Regulation Parameters

2.3.1.1 Parametrizing the Velocity Regulator Set

Parametrizing the velocity regulator set is preferably done with the PI tuning tool using auto tuning. See further details in the TMCL-IDE documentation under the section PI tuning. However, it can be done manually as well without the tool. To parameterize the velocity regulator set properly, do as follows:

1. Set the velocity I parameter to zero.
2. Start the motor by using a medium target velocity (example, 2000rpm).
3. Modify the current P parameter.
   (a) Start from a low value and go to a higher value, until the actual motor speed reaches 80% or 90% of the target velocity.
   (b) The lasting 10% or 20% speed difference can be reduced by slowly increasing the velocity I parameter.

2.4 Velocity Ramp Generator

For a controlled start-up of the motor's velocity, a velocity ramp generator can be activated/deactivated by object 2037_h, sub-object **Enable**. The ramp generator uses the maximal allowed motor velocity (Object $607F_h$), the acceleration (Object 6083_h), and the desired target velocity (Object $60FF_h$) to calculate a ramp generator velocity for the following velocity PI regulator.

2.5 Position Regulation

Based on current and velocity regulators, the TMCM-1690 supports a positioning mode based on encoder (ABN or absolute) or hall sensor position. During positioning, the velocity ramp generator can be activated to enable motor positioning with controlled acceleration or it can be disabled to support motor positioning with maximum allowed speed.

The position regulation uses only one basic parameter: the P regulation parameter

2.5.1 Structure of the Position Regulator

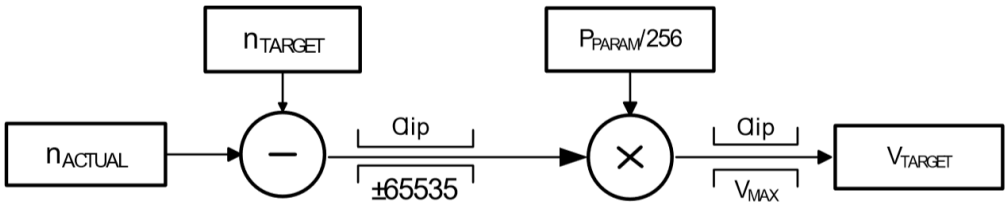


Figure 4: Positioning Regulation (See Parameter Descriptions in Table 4)

Position Regulation Parameters	
Parameter	Description
n_{ACTUAL}	Actual motor position (Object 6064_h)
n_{TARGET}	Target motor position (Object $607A_h$)
P_{PARAM}	Position P parameter (Object 2028_h, sub-object **P**)
V_{MAX}	Max. allowed velocity (Object $607F_h$)
V_{TARGET}	New target velocity for the ramp generator

Table 4: Position Regulation Parameters

2.5.1.1 Parametrizing the Position Regulation

Based on the velocity regulator, only the position regulator P has to be parameterized.
Parametrizing the position regulator is preferably done with the PI tuning tool using auto tuning. See further details in the TMCL-IDE documentation under the section PI tuning. However, it can be done manually as well without the tool. To parameterize the position regulator, do as follows:

1. Disable the velocity ramp generator and set position P parameter to zero.
2. Choose a target position and increase the position P parameter until the motor reaches the target position approximately.

3. Switch on the velocity ramp generator. Based on the maximum positioning velocity (Object $607F_h$) and the acceleration value (Object 6083_h), the ramp generator automatically calculates the slow down point, that is, the point at which the velocity must be reduced to stop at the desired target position.
4. Reaching the target position is signaled by setting the position end flag.

To minimize the time until this flag becomes set, the positioning tolerance MVP target reached distance can be chosen with object 2028_h sub-object **Position reached distance**.

Since the motor typically is assumed not to signal target reached when the target is just passed in a short moment at a high velocity, additionally, the maximum target reached velocity (MVP target reached velocity) can be defined by object 6082_h.

A value of zero for object 6082_h is the most universal, since it implies that the motor stands still at the target. But when a fast rising of the position end flag is desired, a higher value for the MVP target reached velocity parameter saves a lot of time. The best value should be tried out in the actual application.

2.5.2 Correlation of Target Position and the Position End Flag

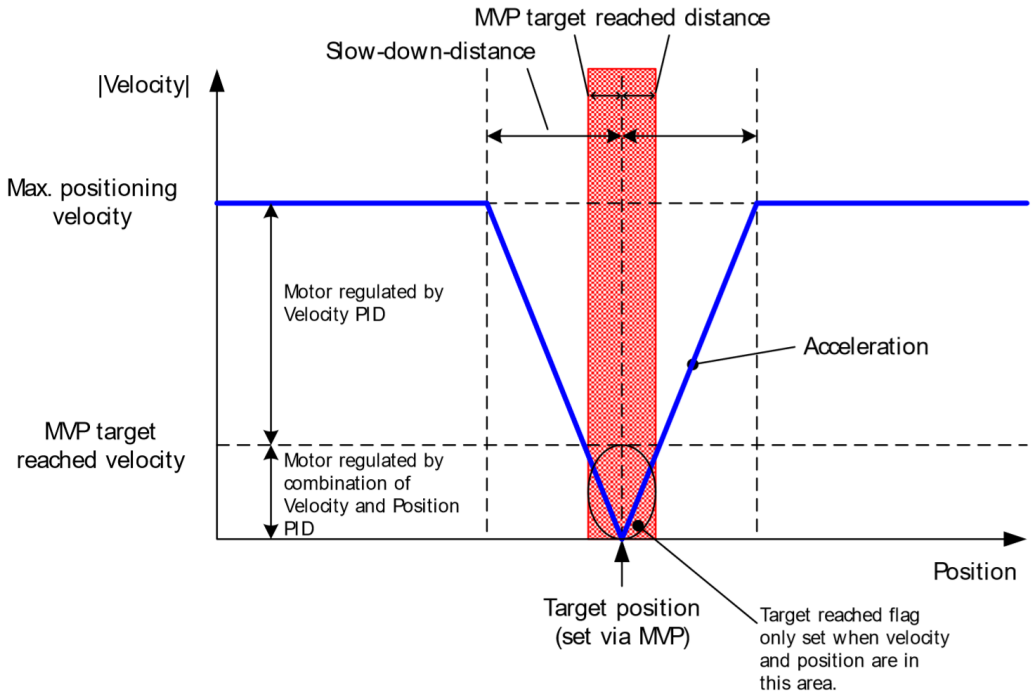


Figure 5: Positioning Algorithm

Depending on motor and mechanics, a low oscillation is normal. This can be reduced to at least +/-1 steps. Without oscillation, the regulation cannot keep the position!

3 Ramps

TMCM-1690 supports two forms of ramps for motion control:

- Linear ramp
- Sine-shaped ramp

3.1 Linear Ramp

A linear ramp or a trapezoidal ramp predicts the acceleration rates by using a constant gradient. This constant gradient results in constant increase in the velocity. The acceleration and velocity are limited by the object 6083_h **(Profile acceleration)** and $607F_h$ **(Max profile velocity)**, respectively. This ramp can be used to perform positioning tasks as well. There are some resonances in the system but for many systems, however, they can be ignored. The linear ramp is selected by setting 0 to object 2003_h sub-object **Ramp type**.

3.2 Sine-Shaped Ramp

In contrast to linear-shaped ramps, the sine-shaped ramps follow a constant increase in the acceleration as it is one order higher. This results in a smoother response with far less resonances in the system. Sine-shaped ramp is selected by setting 1 to object 2003_h sub-object **Ramp type**. Figure 6 gives an impression of how sine-shaped ramp looks like and how it is different from linear ramp. In the figure, desired position Pt is reached with the maximum acceleration and maximum velocity.

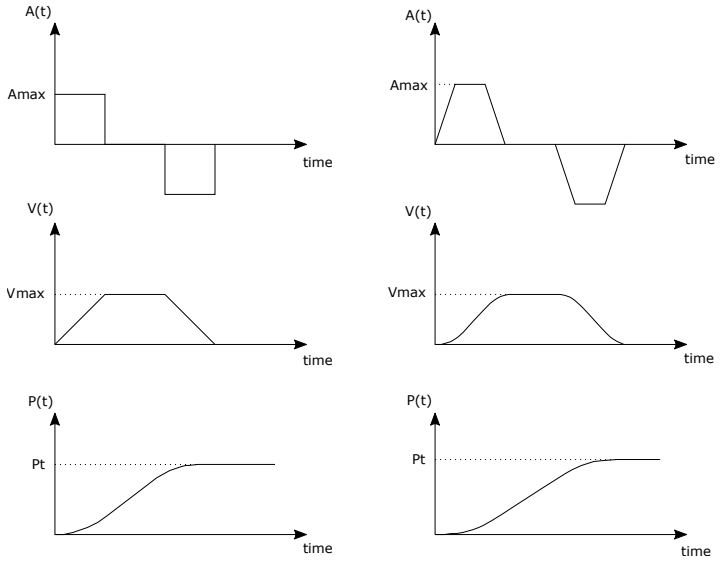


Figure 6: Linear-Shaped Ramp (Left) and Sine-Shaped Ramp (Right)

It can be seen that sine-shaped ramps provide smoother operation as the acceleration is changed linearly compared to abrupt change of acceleration as in the linear ramp. This results from controling a finite amount of jerk at the time when the ramp is accelerating or deccelerating
After the ramp type is selected, the motor can be turned in both velocity and position mode. For turning the motor in velocity mode, the target velocity object $60FF_h$ is set to the desired velocity. The ramp tries

to reach its desired value with fastest acceleration limited by the object 6083_h. For turning the motor with position mode, the target position is set with the object $607A_h$. The ramp is calculated that it tries to reach its desired position with the fastest accelration (which is also limited by object 6083_h) and fastest velocity (limited by maximum velocity object 6081_h).
TMCM-1690 supports all three modes (torque, velocity and position modes) with both linear and sine-shaped ramps. The parameters such as maximum velocity, maximum acceleration, target velocity, and target position can be changed on the fly and the ramp is adjusted accordingly.

4 Module Specific Functions

4.1 Filters

This module supports software filters for five signals, as shown in Figure 7. Each signal can be filtered using either a biquad filter or an averaging filter. Of course, there is the option to not filter at all. The averaging filter averages over the last number of samples, in exponents of 2, up to 2^8. The biquad filter, on the other hand, can be configured as a low-pass filter of the first and the second orders, and also as resonance-anti-resonance filter. As the equation determining the biquad filter is the same, only a different set of coefficients is required to alter its behaviour. See section 4.1.1 for more details.

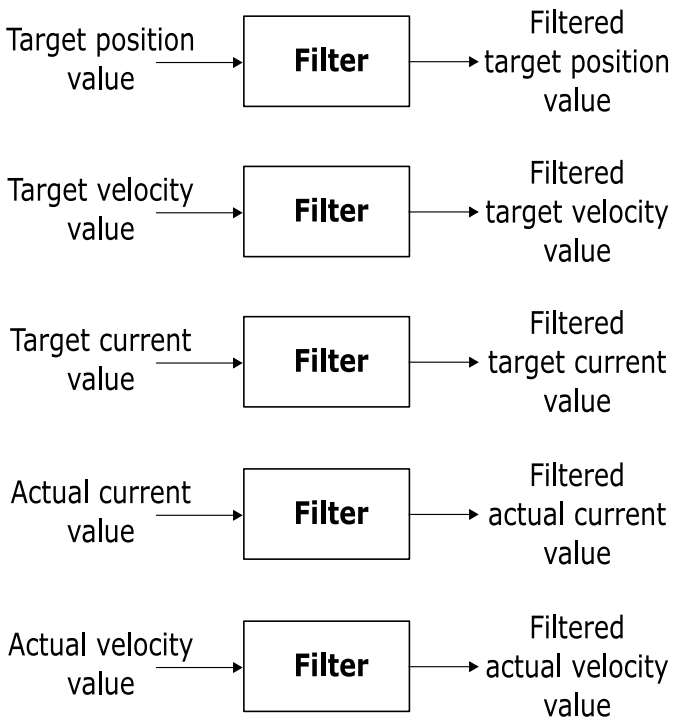


Figure 7: Filter blocks

The filter for the target position value is intended to smoothen position input to the control structure. It is evaluated at every tenth PWM cycle. The filter for the target velocity value is intended to smoothen velocity input from an external controller or the position controller. It is also evaluated at every tenth PWM cycle. The filter for the target torque value is intended to smoothen torque input from an external controller or the velocity controller. It is also evaluated at every tenth PWM cycle. The filter for the actual current value is intended to smoothen the measured actual current value. It is evaluated at every PWM cycle. The filter for the actual velocity value is intended to smoothen the measured actual velocity value. It is evaluated at every tenth PWM cycle. It can be used as a low-pass filter for bandwidth limitation and noise suppression. Moreover, it can be designed to suppress a resonance or anti-resonance. Same statements are correct for all the filters.
There are a total of seven sub-objects for each signal. To set the type of filter for target torque, for example, sub-object **Filter type** in object 2041_h **(Filter target torque)** can be set to values 0, 1, or 2. Where 0 disables the filter, 1 enables the averaging filter and 2 enables the biquad filter. Sub-object **Average filter size** of object 2041_h sets the sample size for the averaging filter. Sub-objects 3 through 7 of object

2041_h correspond to the biquad filter coefficients in Q3.29 format. A similar layout of filter settings is followed by all five signals. See the object table for the filter under consideration.

4.1.1 Biquad Filter (For Future Use)

The biquad filters implemented inside the TMCM-1690 use standard biquad filters (standard IIR filter of second order, Wikipedia Article) in the following structure.

$$\text{Y(n)} = \text{X(n)} \times \text{b\_0} + \text{X(n-1)} \times \text{b\_1} + \text{X(n-2)} \times \text{b\_2} + \text{Y(n-1)} \times \text{a\_1} + \text{Y(n-2)} \times \text{a\_2} \quad (1)$$

In this equation, X(n) is the actual input sample, while Y(n-1) is the filter output of the last cycle. All coefficients are S32 values and are normalized to a Q3.29 format. Take care of correct parametrization of the filter. There is no built-in plausibility or stability check. Biquad state variables are reset when parameters are changed. The TRINAMIC IDE supports parametrization with the *Control Settings* tool.
A standard biquad filter has the following transfer function in the Laplace-Domain:

$$\text{G(s)} = \frac{\text{b\_2\_cont} \times s^2 + \text{b\_1\_cont} \times \text{s} + \text{b\_0\_cont}}{\text{a\_2\_cont} \times s^2 + \text{a\_1\_cont} \times \text{s} + \text{a\_0\_cont}} \quad (2)$$

The transfer function must be transformed to time discrete domain by Z-Transformation and coefficients must be normalized. This is done by the following equations.

$$b\_2\_z = (b\_0\_cont \times T^2 + 2 \times b\_1\_cont \times T + 4 \times b\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \quad (3)$$
$$b\_1\_z = (2 \times b\_0\_cont \times T^2 - 8 \times b\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \quad (4)$$
$$b\_0\_z = (b\_0\_cont \times T^2 - 2 \times b\_1\_cont \times T + 4 \times b\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \quad (5)$$
$$a\_2\_z = (T^2 + 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \quad (6)$$
$$a\_1\_z = (2 \times T^2 - 8 \times a\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \quad (7)$$
$$b\_0 = round(b\_0\_z \times 2^{29}) \quad (8)$$
$$b\_1 = round(b\_1\_z \times 2^{29}) \quad (9)$$
$$b\_2 = round(b\_2\_z \times 2^{29}) \quad (10)$$
$$a\_1 = round(-a\_1\_z \times 2^{29}) \quad (11)$$
$$a\_2 = round(-a\_2\_z \times 2^{29}) \quad (12)$$

While T is the sampling time according to PWM_MAX_COUNT $\times$ 10 ns and variables with index z are auxiliary variables.
A standard second order low-pass filter with given cutoff frequency ω_c and damping factor D has the following transfer function in the Laplace-Domain:

$$G_{LP}(s) = \frac{1}{\frac{1}{\omega_c^2} \times s^2 + \frac{2D}{\omega_c} \times s + 1} \quad (13)$$

Determine filter coefficients with the equations above by comparing coefficients of both transfer functions.

4.2 Mechanical Brake

The TMCM-1690 also supports a mechanical brake that can have a seperate power source. As the brake is operated by MOSFETs on the base board, there is an upper limit to the current it can handle. Thus, provide the brake's electrical resistance and supply voltage to set the upper limit of PWM duty cycle.

The object used for mechanical brake for TMCM-1690 is $205A_h$ **(Mechanical brake)**. For setup, first provide the supply voltage with sub-object **Supply voltage** and brake's resistance with sub-object **Resistance**. Then, provide the duty cycles that the brake has on release with sub-object **Releasing duty cycle** and as hold with sub-object **Holding duty cycle**, while also providing release duration with sub-object **Releasing duration**. Setting sub-object **Enable brake output** enables the brake to be used. Now, the brake can be released by setting sub-object **Release brake**. This causes the brake line to output a PWM signal with release duty cycle for the release duration. After the release duration, the PWM signal goes to holding duty cycle. See Figure 8.

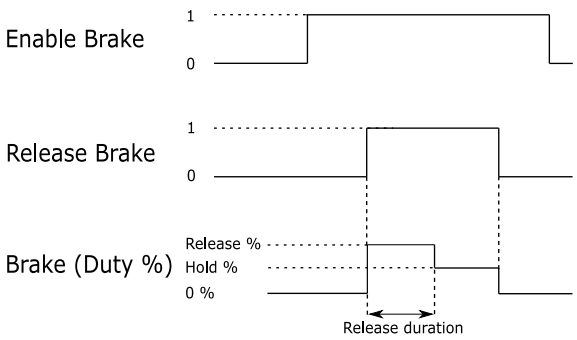


Figure 8: Mechanical Brake Settings

4.3 Brake Chopper

A servo system feeds back energy to the power supply line during deceleration and load control. The energy can lead to a voltage rise on the power supply system if it is not dissipated. The voltage overshoot of a system without brake chopper depends on the motor deceleration time, kinetic energy, and the servo module buffer capacity. The brake chopper dissipates this energy from the system, and thus avoids system damage. TMCM-1690 supports two kinds of brake chopper:

- PWM braking
- Resistive/Shunt braking

4.3.1 PWM Braking

In PWM braking, the motor coils are used to perform the dissipation of the energy from the system. This is done by applying 50% duty cycle on all three phases of the motor. The object used for brake chopper in TMCM-1690 is $205F_h$. For enabling PWM braking, sub-object **Type** is set to 0. Additionally, the voltage limits (sub-object **Voltage limit**) is selected such that the supply voltage is larger than it. After that, the brake is enabled by setting sub-object **Enable**. This results in the extra regenerative energy produced due to slowing down dissipated through the coils of the motor.

4.3.2 Resistive/Shunt Braking

TMCM-1690 provides a continuous motor voltage monitoring as well as brake chopper output. The brake resistor is connected between the supply voltage and brake output. For setup, first provide the supply voltage sub-object **Supply voltage** and brake's resistance (sub-object **Resistance**). Shunt braking is selected by setting sub-object **Type** to 1. The voltage limits and hysteresis are selected using the sub-objects

Voltage limit and **Type**, respectively. Lastly, the brake is enabled by setting sub-object **Enable**. For a full speed ramp stop, the brake resistor should be able to dissipate the complete kinetic energy fed back during deceleration phase. Figure 9 shows an example of brake chopper schematic.

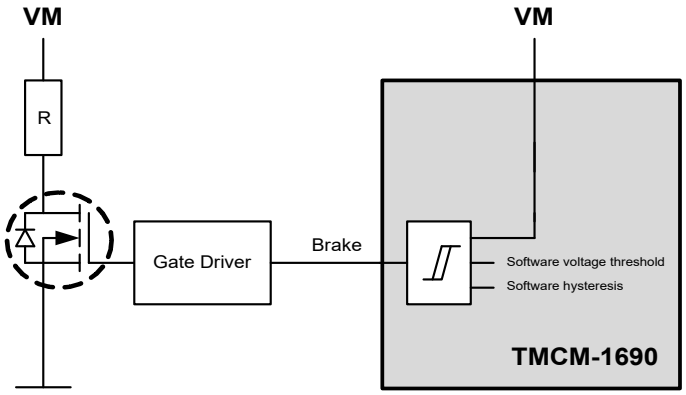


Figure 9: Brake Chopper Example Schematic

4.4 IIT

The IIT monitor checks the energy consumption. The actual current is monitored, and these values are squared and summed up periodically over the configured winding time using a 1ms cycle. If one of the limits gets exceeded during this time, the motor is stopped and the IIT error flag is set. The object corresponding to IIT settings is $202D_h$ **(IIT)**. The IIT error flag can be reset by writing any value to sub-object **Clear exceed flags**. There are two IIT windows (see Figure 10). The first one directly uses the actual current, and the second one uses the actual current divided by $\sqrt{2}$ (less power over longer time). Sub-objects **Limit 1** and **Limit 2** are limits for the two windows. Sub-objects **Sum 1** and **Sum 2** show the actual integration sums.

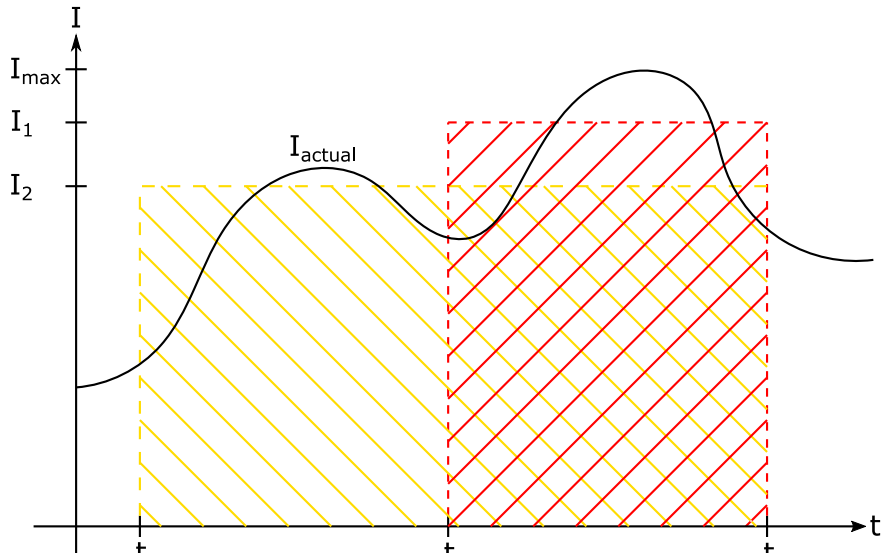


Figure 10: IIT Monitor Windows

4.5 Lower Velocity PI

The TMCM-1690 supports different PI parameters for lower velocities. This provides a better control and smoother operation. It is possible to have a stronger set of PI parameters for the lower velocities and a relatively weaker set for the higher velocities. It is good to have the set of both PI parameters and the velocity to perform the switch (to get the parameters, use the PI tuning tool). The object used for lower velocity PI parameters is $206B_h$ **(Lower velocity PI)** The velocity at which the switch is performed can be set using the sub-object **Switch over speed** (default value is 0). After that, P and I parameters can be set using the sub-objects **Lower velocity P** and **Lower velocity I**, respectively. These are the PI parameters for the velocities less than the switch over velocity. Parameters for higher velocities are set by writing on the standard PI velocity parameters

4.6 General-Purpose Inputs/Outputs

The TMCM-1690 offers versatile General-Purpose Input/Output (GPIO) capabilities, enabling GPIO pins (GPIO2 to GPIO7) to be configured as either an input or an output. By default, all GPIOs are set as inputs. To switch each GPIO state between input and output, modify object 0x2701.
Object 0x2701 is a bit vector, where each bit determines if the corresponding digital line is an input or an output. A bit that is clear means input, and a bit that is set means output.

GPIO Pin	Bit Setting	Value
GPIO2	2	4
GPIO3	3	8
GPIO4	4	16
GPIO5	5	32
GPIO6	6	64
GPIO7	7	128

Table 5: GPIO Pin Configuration

By adjusting these bits, it is possible to control the input/output configuration of each GPIO pin on the TMCM-1690. For example, setting object 0x2701 to 20 programs GPIO2 and GPIO4 as outputs and the rest of the GPIOs as inputs.

5 Communication

5.1 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 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 through the CAN network with peer service object(s) using a protocol.

The application and 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 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 6: Service Primitives

A service type defines the primitives 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 7: Service Types

5.2 Network Management (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 through the state machine. The NMT state machine consists of the states shown in Figure 11.

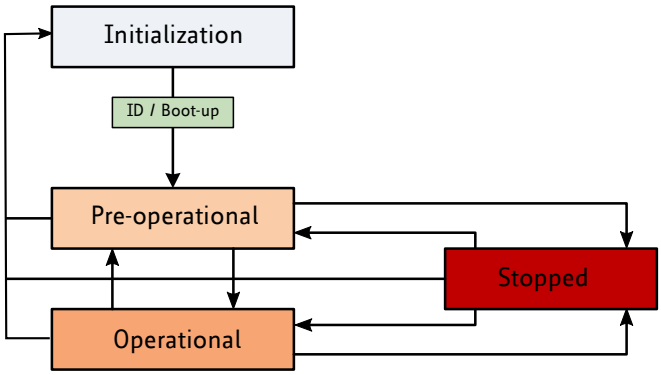


Figure 11: NMT State Machine

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 it is ready to work. A device that stays in pre-operational state may start to transmit SYNC-, time stamp-, or heartbeat messages. In contrast to the PDO communication that is disabled in this state, the device can communicate through SDO.

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

A device 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 (example, to start and stop the device), and 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 that 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 in OPERATION_ENABLED state switch to FAULT state.

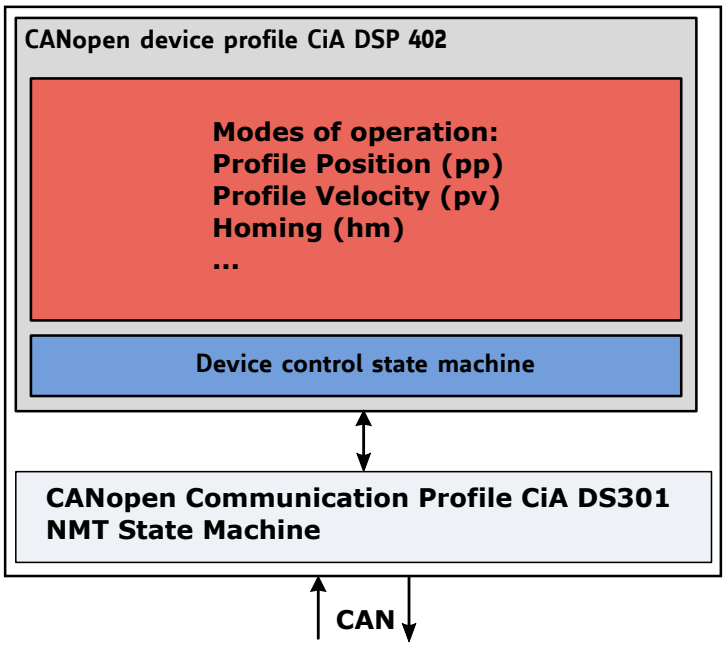


Figure 12: Communication Architecture

5.3 Device Model

A CANopen device consists of the following parts:

- *Communication:* This function unit provides the communication objects and the appropriate functionality to transport data items through the underlying network structure.
- *Object Dictionary:* The object dictionary is a collection of all the data items that 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.

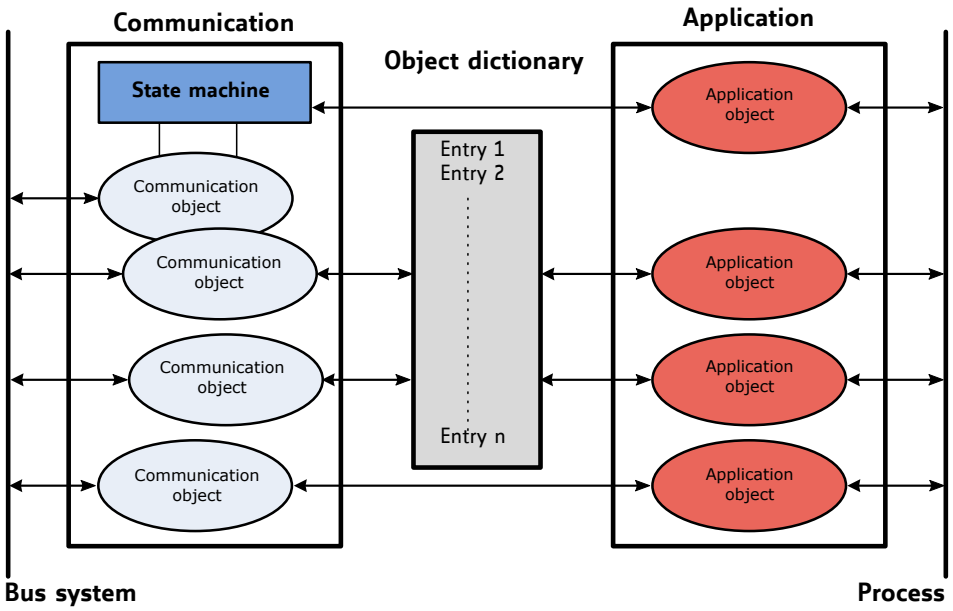


Figure 13: Device Model

5.4 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 through the network in an ordered predefined 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 8:

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

Table 8: 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 TMCM-1690.

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 through the network. They describe the device parameters and the device functionality of the device profile.

6 Communication Area

The communication area contains all objects that define the communication parameters of the CANopen device according to the DS301 standard.

6.1 Detailed Object Specifications

6.1.1 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 that describes the device profile used and a second 16-bit field that provides additional information about optional functionality of the device.

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

Table 9: Object Description (1000_h)

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

Table 10: Entry Description (1000_h)

6.1.2 Object 1001_h: Error Register

This object contains error information. The CANopen device maps internal errors into object 1001_h. It is part of an emergency object.

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

Table 11: Object Description (1001_h)

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

Table 12: 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 13: Error Register Bits

6.1.3 Object 1005_h: COB-ID SYNC

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 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 14: Value Definition (1005_h)

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

Table 15: Object Description (1005_h)

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

Table 16: Entry Description (1005_h)

6.1.4 Object 1008_h: Manufacturer Device Name

This object contains the name of the device given by the manufacturer.

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

Table 17: Object Description (1008_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	—	TMCM-1690

Table 18: Entry Description (1008_h)

6.1.5 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 19: Object Description (1009_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	—	Depends on device. Example: 1.00

Table 20: Entry Description (1009_h)

6.1.6 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 21: Object Description ($100A_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	—	Depends on device. Example: 1.00

Table 22: Entry Description ($100A_h$)

6.1.7 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.

The TMCM-1690 module supports saving of the following parameter groups:

- Subindex 1: save all parameters.
- Subindex 2: save communication parameters.
- Subindex 3: save device profile parameters (not used).
- Subindex 4: save motor 0 parameters.
- Subindex 5: save device parameters (other non axis-related parameters).

Note To avoid storage of parameters by mistake, storage is only executed when a specific signature is written to the appropriate subindex. 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 subindex, 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 subindex 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
0	Highest supported sub-index	ro	no	UNSIGNED8	5
1	Save all parameters	rw	no	UNSIGNED32	1
2	Save communication parameters	rw	no	UNSIGNED32	1
3	Save device profile parameters	rw	no	UNSIGNED32	0
4	Save motor 0 parameters	rw	no	UNSIGNED32	1
5	Save device parameters	rw	no	UNSIGNED32	1

Table 25: Entry Description (1010_h)

6.1.8 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.

The TMCM-1690 module supports restoring of the following parameter groups:

- Subindex 1: restore all parameters.
- Subindex 2: restore communication parameters.
- Subindex 3: restore device profile parameters (not used).
- Subindex 4: restore motor 0 parameters.
- Subindex 5: restore device parameters (other non axis-related parameters).

Note To avoid restoring the parameters by mistake, restoring is only executed when a specific signature is written to the appropriate subindex. 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 subindex, 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 subindex provides information if it is possible to restore the parameter group. It reads 1 if yes and 0 if no.

After the default values are restored, they become active after the next reset or power cycle of the TMCM-1690.

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
0	Highest supported sub-index	ro	no	UNSIGNED8	5
1	Restore all parameters	rw	no	UNSIGNED32	1
2	Restore communication parameters	rw	no	UNSIGNED32	1
3	Restore device profile parameters	rw	no	UNSIGNED32	0
4	Restore motor 0 parameters	rw	no	UNSIGNED32	1
5	Restore device parameters	rw	no	UNSIGNED32	1

Table 28: Entry Description (1011_h)

6.1.9 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)

6.1.10 Object 1015_h: Inhibit Time EMCY

The inhibit time for the EMCY message can be adjusted through this entry. The time has to be a multiple of 100µs.

Object Description			
Index	Name	Object Type	Data Type
1015_h	Inhibit Time EMCY	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)

6.1.11 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	Array	UNSIGNED32

Table 34: Object Description (1016_h)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
0	Number of entries	ro	no	UNSIGNED8	1
1	Consumer Heartbeat Time1	rw	no	UNSIGNED32	0

Table 35: Entry Description (1016_h)

6.1.12 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)

6.1.13 Object 1018_h: Identity Object

The object 1018_h contains general information about the device:

- The vendor ID (subindex 01_h) contains a unique value allocated to each manufacturer. The vendor ID of ADI Trinamic is 286_h.
- The manufacturer specific product code (subindex 2_h) identifies a specific device version.
- The manufacturer specific revision number (subindex 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
0	Number of entries	ro	no	0…3	3
1	Vendor ID	ro	no	UNSIGNED32	0286_h
2	Product code	ro	no	UNSIGNED32	1690
3	Revision number	ro	no	UNSIGNED32	e.g. 20003_h for version 2.3
4	Serial number	ro	no	UNSIGNED32	depends on module

Table 39: Entry Description (1018_h)

6.1.14 Object 1029_h: Error Behavior

If a device failure is detected in operational state, the device can be configured to alternatively enter 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 Behavior	Array	UNSIGNED8

Table 40: Object Description (1029_h)

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

Table 41: Entry Description (1029_h)

6.1.15 Objects 1400_h to 1403_h: Receive PDO Communication Parameter

This object contains the communication parameters for the RPDOs the device is able to receive. The sub-index 0 contains the number of valid entries within the communication record. Its value normally is 2, as this object consists of two other entries.

Subindex 1 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 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.

Subindex 2 contains the transmission type of the RPDO. This can be FF_h or FE_h for event-driven, or 1…240 for synchronous (1 means the PDO is processed with every SYNC message, and 4, for example, means the PDO is processed with every fourth SYNC message). Other values are not supported.

Object Description			
Index	Name	Object Type	Data Type
1400_h to 1403_h	Receive PDO Communication 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				
Subindex	Description	Access	Value Range	Default Value
0	Largest subindex supported	ro	2	2
1	COB-ID used by PDO	rw	UNSIGNED32	Index 1400_h: 200_h + Node-ID Index 1401_h: 300_h + Node-ID Index 1402_h: 400_h + Node-ID Index 1403_h: 500_h + Node-ID
2	Transmission Type	rw	UNSIGNED8	Index 1400_h: FF_h Index 1401_h: FF_h Index 1402_h: FF_h Index 1403_h: FE_h

Table 43: Entry Description (1400_h)

6.1.16 Objects 1600_h to 1603_h: Receive PDO Mapping Parameter

These objects contain the mapping parameters for the RPDOs the device is able to receive. The subindex 0 contains the number of valid entries within the mapping record. This number of entries is also the number of the application variables received with the corresponding RPDO. The sub-indices from 1 to the number of entries contain the information about the mapped application variables. These entries describe the PDO contents by their index, subindex, and length.

Object Description			
Index	Name	Object Type	Data Type
1600_h to 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				
Subindex	Description	Access	Value Range	Default Value
0	Number of mapped application objects in PDO	rw	0...3	Index 1600_h: 1 Index 1601_h: 2 Index 1602_h: 2 Index 1603_h: 2
1	Mapping Entry 1	rw	UNSIGNED32	Index 1600_h: 60400010_h Index 1601_h: 60400010_h Index 1602_h: 60400010_h Index 1603_h: 60400010_h
2	Mapping Entry 2	rw	UNSIGNED32	Index 1600_h: 0 Index 1601_h: 60600008_h Index 1602_h: $607A0020_h$ Index 1603_h: $60FF0020_h$
3	Mapping Entry 3	rw	UNSIGNED32	Index 1600_h: 0_h Index 1601_h: 0_h Index 1602_h: 0_h 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 6.1.15). Then, set its number of mapped PDO entries to zero (subindex 0 of the appropriate PDO mapping object). Now, the mapppings themselves 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.

6.1.17 Objects 1800_h to 1803_h: Transmit PDO Communication Parameter

This object contains the communication parameters for the TPDOs the device is able to transmit. The subindex 0 contains the number of valid entries within the communication record. Its value normally is 5, as this object consists of five other entries.

Subindex 1 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 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.

Subindex 2 contains the transmission type of the RPDO. This can be FF_h or FE_h for event-driven or 1...240 for synchronous (1 means that the PDO is sent with every SYNC message, and 4, for example, means that the PDO is sent with every fourth SYNC message). Other values are not supported.

Subindex 3 contains the inhibit time, given in units of 0.1ms. After a TPDO is sent, it is not sent again before the inhibit time has elapsed.

Subindex 4 is not used.

Subindex 5 contains the event timer value in milliseconds. When this is set to a value greater than 0, the TPDO is sent repeatedly each time the event timer has elapsed. It is also sent when the value has changed before the event timer has elapsed, but not before the inhibit time has elapsed.

Object Description			
Index	Name	Object Type	Data Type
1800_h to 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				
Subindex	Description	Access	Value Range	Default Value
0	Largest subindex supported	ro	5	5
1	COB-ID	rw	UNSIGNED32	Index 1800_h: 180_h + Node-ID Index 1801_h: 280_h + Node-ID Index 1802_h: 380_h + Node-ID Index 1803_h: 480_h + Node-ID
2	Transmission Type	rw	UNSIGNED8	Index 1800_h: FF_h Index 1801_h: FF_h Index 1802_h: 01_h Index 1803_h: 01_h
3	Inhibit Time	rw	UNSIGNED16	0
4	Compatibility Entry	ro	UNSIGNED8	0
5	Event Timer	rw	UNSIGNED16	0

Table 47: Entry Description (1800_h)

6.1.18 Objects $1A00_h$ to $1A03_h$: Transmit PDO Mapping Parameter

These objects contain the mapping parameters for the TPDOs the device is able to transmit. The subindex 0 contains the number of valid entries within the mapping record. This number of entries is also the number of the application variables transmitted with the corresponding TPDO. The subindices from 1 to the number of entries contain the information about the mapped application variables. These entries describe the PDO contents by their index, subindex, and length.

Object Description			
Index	Name	Object Type	Data Type
$1A00_h$ to $1A03_h$	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				
Subindex	Description	Access	Value Range	Default Value
0	Number of mapped application objects in PDO	rw	0...3	Index $1A00_h$: 1 Index $1A01_h$: 2 Index $1A02_h$: 2 Index $1A03_h$: 2
1	Mapping Entry 1	rw	UNSIGNED32	Index $1A00_h$: 60410010_h Index $1A01_h$: 60410010_h Index $1A02_h$: 60410010_h Index $1A03_h$: 60410010_h
2	Mapping Entry 2	rw	UNSIGNED32	Index $1A00_h$: 0 Index $1A01_h$: 60610008_h Index $1A02_h$: 60640020_h Index $1A03_h$: $606C0020_h$
3	Mapping Entry 3	rw	UNSIGNED32	Index $1A00_h$: 0_h Index $1A01_h$: 0_h Index $1A02_h$: 0_h 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 6.1.17). Then, set its number of mapped PDO entries to zero (subindex 0 of the appropriate PDO mapping object). Now, the mappings themselves 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.

7 Manufacturer Specific Area

The manufacturer segment contains manufacturer specific objects. These objects control the special features of the Trinamic Motion Control device TMCM-1690.

7.1 Detailed Object Specifications

7.1.1 Object 2000_h: Motor Settings

This object provides information about motor settings that can be deduced from the data sheet.

Object Description			
Index	Name	Object Type	Data Type
2000_h	Motor settings	Variable	Record

Table 50: Object Description (2000_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Motor type	no	0	1	0	—	RW
2	Motor family	no	0	1	0	—	RW
3	Pole pairs	no	0	255	4	—	RW
4	Pole pair distance	no	1	65535	10	μm	RW
5	Nominal current	no	0	10000	3470	mA	RW
6	Peak current	no	0	10000	10000	mA	RW
7	Line to line resistance	no	1	65535	720	m Ω	RW
8	Line to line inductance	no	1	65535	1200	μH	RW
9	Torque constant	no	1	65535	30	mNm/A	RW
10	Inertia	no	1	65535	480	gcm^2	RW

Table 51: Entry Description (2000_h)

Parameter	Description
Motor type	Select a motor type connected to the module 0. 3-phase BLDC 1. DC motor
Motor family	Select the motor family 0. Rotary 1. Linear

Table 52: Parameter Description (2000_h)

7.1.2 Object 2003_h: Drive Settings

This object is used to configure the drive settings. It has parameters such as ramp type, commutation mode, sensors, and maximum current to initialize the drive.

Object Description			
Index	Name	Object Type	Data Type
2003_h	Drive settings	Variable	Record

Table 53: Object Description (2003_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Ramp type	no	0	1	0	—	RW
2	Motor direction	no	0	1	0	—	RW
3	Commutation mode	no	0	5	0	—	RW
4	Peripherals	no	0	63	34	—	RW
5	Current sensor selection	no	0	0	0	—	RO
6	Maximum current	no	0	10000	4000	mA	RW
7	PWM scheme	no	0	1	0	—	RW
8	PWM frquency	no	0	5	0	—	RW
9	PWM dead time	no	0	255	1	ms	RW

Table 54: Entry Description (2003_h)

Parameter	Description
Ramp type	Select a ramp type between linear and s shaped ramp 0. Linear ramp 1. S-shaped ramp
Motor direction	Used for reversing motor shaft direction 0. Rotate clockwise 1. Rotate counter clockwise
Commutation mode	Select commutation mode based on the sensor used 0: Disabled 1: Open loop 2: Digital hall (FOC) 3: Digital hall (Block) 4: ABN encoder 5: Absolute encoder
PWM scheme	Select the PWM scheme 0. Center alligned 1. Down counting
PWM frequency	Select the desired PWM frequency(Hz) 0. 20000 1. 40000 2. 60000 3. 80000 4. 100000 5. 120000

Table 55: Parameter Description (2003_h)

7.1.3 Object 2005_h: ADC Settings

Using this object, the ADC offsets for the coil current measurement can be configured. This is necessary for each new motor type.

Object Description			
Index	Name	Object Type	Data Type
2005_h	ADC settings	Variable	Record

Table 56: Object Description (2005_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	ADC_u_raw	no	0	4095	0	—	RO
2	ADC_v_raw	no	0	4095	0	—	RO
3	ADC_w_raw	no	0	4095	0	—	RO
4	ADC_u_offset	no	0	4095	2047	—	RW
5	ADC_v_offset	no	0	4095	2047	—	RW
6	ADC_w_offset	no	0	4095	2047	—	RW
7	ADC_u	no	-32768	32767	0	—	RO
8	ADC_v	no	-32768	32767	0	—	RO
9	ADC_w	no	-32768	32767	0	—	RO

Table 57: Entry Description (2005_h)

7.1.4 Object $200A_h$: Open Loop Settings

This object deals with the open loop settings such as commutation angle, current velocity, and position.

Object Description			
Index	Name	Object Type	Data Type
$200A_h$	Open loop settings	Variable	Record

Table 58: Object Description ($200A_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Commutation angle	no	-32768	32767	0	—	RO
2	Current	no	0	10000	2000	mA	RW
3	Velocity	no	-2147483648	2147483648	0	—	RO
4	Position	no	-2147483648	2147483648	0	—	RW

Table 59: Entry Description ($200A_h$)

7.1.5 Object $200F_h$: Digital Hall Settings

This object relates to the digital hall sensor settings. It has parameters to initialize the hall sensor and run the motor using hall sensor as commutation mode.

Object Description			
Index	Name	Object Type	Data Type
$200F_h$	Digital hall settings	Variable	Record

Table 60: Object Description ($200F_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Sector offset	no	0	5	0	—	RW
2	Direction	no	0	1	0	—	RW
3	Interpolation	no	0	1	0	—	RW
4	Offset	no	-32768	32767	0	—	RW
5	Inputs	no	0	7	0	—	RO
6	Identify trigger configuration	no	0	2	0	—	RW
7	Commutation angle	no	-32768	32767	0	—	RO
8	Open Loop angle difference	no	-32768	32767	0	—	RO
9	Velocity	no	-2147483648	2147483648	0	—	RO
10	Position	no	-2147483648	2147483648	0	—	RW

Table 61: Entry Description ($200F_h$)

Parameter	Description
Sector Offset	Select the sector offset for configuring hall sensor (degrees) 0. 0 1. 60 2. 120 3. 180 4. 240 5. 300
Direction	Hall sensor direction 0. Standard 1. Inverted
Interpolation	Hall sensor interpolation 0. Off 1. On
Identify trigger configuration	Select the method for identifying trigger configuration of hall sensor 0: Standby 1: Direction estimation 2: Sector offset estimation

Table 62: Parameter Description ($200F_h$)

7.1.6 Object 2014_h: ABN Encoder Settings

This object relates to the ABN encoder settings. It has parameters to initialize the ABN encoder and run the motor using ABN encoder as commutation mode.

Object Description			
Index	Name	Object Type	Data Type
2014_h	ABN encoder settings	Variable	Record

Table 63: Object Description (2014_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Steps	no	0	1050000	4096	—	RW
2	Direction	no	0	1	0	—	RW
3	Offset	no	0	1050000	0	—	RW
4	Clear on null	no	0	1	0	—	WO
5	Clear once	no	0	1	0	—	WO
6	Init mode	no	0	5	0	—	RW
7	Init state	no	0	3	0	—	RO
8	Init delay	no	0	10000	1000	ms	RW
9	Init velocity	no	-200000	200000	0	rpm	RW
10	Inputs	no	0	7	0	—	RO
11	Encoder value	no	-2147483648	2147483648	0	—	RO
12	Sensor side	no	0	1	0	—	RW
13	Commutation angle	no	-32768	32767	0	—	RO
14	Open loop angle difference	no	-32768	32767	0	—	RO
15	Velocity	no	-2147483648	2147483648	0	—	RO
16	Position	no	-2147483648	2147483648	0	—	RW
17	Linear resolution	no	1	65535	500	nm	RW

Table 64: Entry Description (2014_h)

Parameter	Description
Direction	ABN encoder direction 0. Standard 1. Inverted
Init mode	Select the method for initialization of ABN encoder 0. Estimate offset 1. Estimate offset (Shake) 2. Use offset 3. Use hall 4. Estimate offset (OL) 5. Estimate offset (Linear)
Sensor side	Select where the encoder is connected 0: Disabled 1: Motor side

Table 65: Parameter Description (2014_h)

7.1.7 Object 2015_h: ABN Encoder 2 Settings

This object relates to the second ABN encoder settings. It has parameters to initialize the second ABN encoder and run the motor with the second ABN encoder as a backup encoder.

Object Description			
Index	Name	Object Type	Data Type
2015_h	ABN encoder 2 settings	Variable	Record

Table 66: Object Description (2015_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Steps	no	0	65535	4096	—	RW
2	Direction	no	0	1	0	—	RW
3	Inputs	no	0	7	0	—	RO
4	Sensor side	no	0	2	0	—	RW
5	Commutation angle	no	-32768	32767	0	—	RO
6	Velocity	no	-2147483648	2147483648	0	—	RO
7	Position	no	-2147483648	2147483648	0	—	RW

Table 67: Entry Description (2015_h)

Parameter	Description
Direction	ABN 2 encoder direction 0. Standard 1. Inverted
Sensor side	Select where the encoder is connected 0: Disabled 1: Motor side 2: Load side

Table 68: Parameter Description (2015_h)

7.1.8 Object 2019_h: Absolute Encoder Settings

This object relates to the absolute encoder settings. It has parameters to initialize the absolute encoder and run the motor using the absolute encoder as commutation mode.

Object Description			
Index	Name	Object Type	Data Type
2019_h	Absolute encoder settings	Variable	Record

Table 69: Object Description (2019_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Type	no	0	2	0	—	RW
2	Init mode	no	0	2	0	—	RW
3	Direction	no	0	1	0	—	RW
4	Offset	no	-32768	32767	0	—	RW
5	Sensor side	no	0	2	0	—	RW
6	Commutation angle	no	-32768	32767	0	—	RO
7	Open loop angle difference	no	-32768	32767	0	—	RO
8	Velocity	no	-2147483648	2147483648	0	—	RO
9	Position	no	-2147483648	2147483648	0	—	RW

Table 70: Entry Description (2019_h)

Parameter	Description
Type	Select the absolute encoder connected 0. Disabled 1. AM4096 2. IC MU_150
Direction	Absolute encoder direction 0. Standard 1. Inverted
Init mode	Select the method for initialization of absolute encoder 0. Estimate offset 1. Estimate offset (shake) 2. Use offset
Sensor side	Select where the encoder is connected 0: Disabled 1: Motor side 2: Load side

Table 71: Parameter Description (2019_h)

7.1.9 Object $201E_h$: Torque Mode

Object Description			
Index	Name	Object Type	Data Type
$201E_h$	Torque mode settings	Variable	Record

Table 72: Object Description ($201E_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Target flux	no	-10000	10000	0	mA	RW
2	Actual flux	no	-2147483648	2147483647	0	mA	RO
3	P	no	0	32767	300	—	RW
4	I	no	0	32767	600	—	RW
5	Torque PI error sum	no	-2147483648	2147483647	0	—	RO
6	Flux PI error sum	no	-2147483648	2147483647	0	—	RO
7	Torque PI error	no	-2147483648	2147483647	0	—	RO
8	Flux PI error	no	-2147483648	2147483647	0	—	RO

Table 73: Entry Description ($201E_h$)

Parameter	Description
P	P parameter for the torque PI controller.
I	I parameter for the torque PI controller.

Table 74: Parameter Description ($201E_h$)

7.1.10 Object 2023_h: Velocity Mode

Object Description			
Index	Name	Object Type	Data Type
2023_h	Velocity mode	Variable	Record

Table 75: Object Description (2023_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Sensor selection	no	0	4	0	—	RW
2	Unit selection	no	0	1	0	—	RW
3	Halted velocity	no	0	200000	0	rpm	RW
4	P	no	0	32767	600	—	RW
5	I	no	0	32767	300	—	RW
6	PI error sum	no	-2147483648	2147483647	0	—	RO
7	PI error	no	-2147483648	2147483647	0	—	RO

Table 76: Entry Description (2023_h)

Parameter	Description
P	P parameter for the velocity PI controller.
I	I parameter for the velocity PI controller.
Sensor selection	Select a commutation mode that fits best to the motor's sensors. 0: Same as commutation 1: Hall sensor 2. ABN encoder 3. ABN encoder 2 4. Absolute encoder
Unit selection	Select mechanical or electrical velocity unit. 0. Mechanical rpm 1. Electrical rpm
Halted velocity	If the actual velocity is below this value, the motor halted flag is set.

Table 77: Parameter Descritption (2023_h)

7.1.11 Object 2028_h: Position Mode

Object Description			
Index	Name	Object Type	Data Type
2028_h	Position mode	Variable	Record

Table 78: Object Description (2028_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Sensor selection	no	0	4	0	—	RW
2	Position reached distance	no	0	100000	0	—	RW
3	P	no	0	32767	20	—	RW
4	PI error	no	-2147483648	2147483647	0	—	RO

Table 79: Entry Description (2028_h)

Parameter	Description
P	P parameter for the velocity PI controller.
Sensor selection	Select a commutation mode that fits best to the motor's sensors. 0: Same as commutation 1: Hall sensor 2. ABN encoder 3. ABN encoder 2 4. Absolute encoder

Table 80: Parameter Description (2028_h)

7.1.12 Object $202B_h$: Gearbox

This object deals with the gearbox and is used to include a gear ratio at the output. This can be rotary to rotary or rotary to linear.

Object Description			
Index	Name	Object Type	Data Type
$202B_h$	Gearbox	Variable	Record

Table 81: Object Description ($202B_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Transmission type	no	0	1	0	—	RW
2	Motor displacement	no	1	2147483647	1	—	RW
3	Load displacement	no	1	2147483647	1	—	RW
4	Invert direction	no	0	1	0	—	RW

Table 82: Entry Description ($202B_h$)

Parameter	Description
Transmission type	Select the transmission type from rotary to linear or rotary 0: Rotary to rotary 1: Rotary to linear
Invert direction	Use this sub object to invert the direction of gearbox 0. Off 1. On

Table 83: Parameter Description ($202B_h$)

7.1.13 Object $202D_h$: IIT

Object Description			
Index	Name	Object Type	Data Type
$202D_h$	IIT	Variable	Record

Table 84: Object Description ($202D_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Thermal winding time constant 1	no	1000	60000	30000	ms	RW
2	Limit 1	no	0	54000000	10300000	—	RW
3	Sum 1	no	0	4294967295	0	—	RO
4	Thermal winding time constant 2	no	1000	60000	30000	ms	RW
5	Limit 2	no	0	54000000	10300000	—	RW
6	Sum 2	no	0	4294967295	0	—	RO
7	Clear exceeded flags	no	0	1	0	—	RW

Table 85: Entry Description ($202D_h$)

7.1.14 Object 2032_h: Windows

This object is used to set the position and velocity windows and to clear the windows in case of an error.

Object Description			
Index	Name	Object Type	Data Type
2032_h	Windows	Variable	Record

Table 86: Object Description (2032_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Velocity window	no	0	200000	500	—	RW
2	Clear velocity window error	no	0	1	0	—	RW
3	Position window	no	0	2147483647	1638400	—	RW
4	Clear position window error	no	0	1	0	—	RW

Table 87: Entry Description (2032_h)

7.1.15 Object 2037_h: Ramp

Object Description			
Index	Name	Object Type	Data Type
2037_h	Ramp	Variable	Record

Table 88: Object Description (2037_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Enable	no	0	1	0	—	RW
2	Enable velocity feed forward	no	0	1	0	—	RW
3	Acceleration	no	0	1000000	0	rpm/s	RO
4	Velocity	no	-200000	200000	0	rpm	RO
5	Position	no	-2147483648	2147483647	0	—	RO

Table 89: Entry Description (2037_h)

7.1.16 Object $203C_h$: Homing

This object deals with the homing routine.

Object Description			
Index	Name	Object Type	Data Type
$203C_h$	Homing	Variable	Record

Table 90: Object Description ($203C_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	State	no	0	255	0	—	RO
2	Position offset CW	no	0	4294967295	40000	—	RW
3	Position offset CCW	no	0	4294967295	40000	—	RW
4	Current threshold	no	0	10000	2000	mA	RW
5	Teach position limit	no	0	3	0	—	RW

Table 91: Entry Description ($203C_h$)

7.1.17 Object 2041_h: Filter Target Torque

This object is used to configure the filter for target torque.

Object Description			
Index	Name	Object Type	Data Type
2041_h	Filter target torque	Variable	Record

Table 92: Object Description (2041_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Filter type	no	0	2	1	—	RW
2	Average filter size	no	1	8	1	—	RW
3	Biquad coeff a1	no	-2147483648	2147483647	0	—	RW
4	Biquad coeff a2	no	-2147483648	2147483647	0	—	RW
5	Biquad coeff b0	no	-2147483648	2147483647	0	—	RW
6	Biquad coeff b1	no	-2147483648	2147483647	0	—	RW
7	Biquad coeff b2	no	-2147483648	2147483647	0	—	RW

Table 93: Entry Description (2041_h)

Parameter	Description
Filter type	Select the filter type 0: Disabled 1: Average 2. Biquad

Table 94: Parameter Description (2041_h)

7.1.18 Object 2046_h: Filter Actual Torque

This object is used to configure the filter for actual torque.

Object Description			
Index	Name	Object Type	Data Type
2046_h	Filter actual torque	Variable	Record

Table 95: Object Description (2046_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Filter type	no	0	2	1	—	RW
2	Average filter size	no	1	8	1	—	RW
3	Biquad coeff a1	no	-2147483648	2147483647	0	—	RW
4	Biquad coeff a2	no	-2147483648	2147483647	0	—	RW
5	Biquad coeff b0	no	-2147483648	2147483647	0	—	RW
6	Biquad coeff b1	no	-2147483648	2147483647	0	—	RW
7	Biquad coeff b2	no	-2147483648	2147483647	0	—	RW

Table 96: Entry Description (2046_h)

Parameter	Description
Filter type	Select the filter type 0: Disabled 1: Average 2. Biquad

Table 97: Parameter Description (2046_h)

7.1.19 Object $204B_h$: Filter Target Velocity

This object is used to configure the filter for target velocity.

Object Description			
Index	Name	Object Type	Data Type
$204B_h$	Filter target velocity	Variable	Record

Table 98: Object Description ($204B_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Filter type	no	0	2	1	—	RW
2	Average filter size	no	1	8	1	—	RW
3	Biquad coeff a1	no	-2147483648	2147483647	0	—	RW
4	Biquad coeff a2	no	-2147483648	2147483647	0	—	RW
5	Biquad coeff b0	no	-2147483648	2147483647	0	—	RW
6	Biquad coeff b1	no	-2147483648	2147483647	0	—	RW
7	Biquad coeff b2	no	-2147483648	2147483647	0	—	RW

Table 99: Entry Description ($204B_h$)

Parameter	Description
Filter type	Select the filter type 0: Disabled 1: Average 2. Biquad

Table 100: Parameter Description ($204B_h$)

7.1.20 Object 2050_h: Filter Actual Velocity

This object is used to configure the filter for actual velocity.

Object Description			
Index	Name	Object Type	Data Type
2050_h	Filter actual velocity	Variable	Record

Table 101: Object Description (2050_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Filter type	no	0	2	1	—	RW
2	Average filter size	no	1	8	1	—	RW
3	Biquad coeff a1	no	-2147483648	2147483647	0	—	RW
4	Biquad coeff a2	no	-2147483648	2147483647	0	—	RW
5	Biquad coeff b0	no	-2147483648	2147483647	0	—	RW
6	Biquad coeff b1	no	-2147483648	2147483647	0	—	RW
7	Biquad coeff b2	no	-2147483648	2147483647	0	—	RW

Table 102: Entry Description (2050_h)

Parameter	Description
Filter type	Select the filter type 0: Disabled 1: Average 2. Biquad

Table 103: Parameter Description (2050_h)

7.1.21 Object 2055_h: Filter Target Position

This object is used to configure the filter for target position.

Object Description			
Index	Name	Object Type	Data Type
2055_h	Filter target position	Variable	Record

Table 104: Object Description (2055_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Filter type	no	0	2	1	—	RW
2	Average filter size	no	1	8	1	—	RW
3	Biquad coeff a1	no	-2147483648	2147483647	0	—	RW
4	Biquad coeff a2	no	-2147483648	2147483647	0	—	RW
5	Biquad coeff b0	no	-2147483648	2147483647	0	—	RW
6	Biquad coeff b1	no	-2147483648	2147483647	0	—	RW
7	Biquad coeff b2	no	-2147483648	2147483647	0	—	RW

Table 105: Entry Description (2055_h)

Parameter	Description
Filter type	Select the filter type 0: Disabled 1: Average 2. Biquad

7.1.22 Object $205A_h$: Mechanical Brake

This object is used to set up and use the mechanical brake.

Object Description			
Index	Name	Object Type	Data Type
$205A_h$	Mechanical brake	Variable	Record

Table 106: Object Description ($205A_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Release brake	no	0	1	0	—	RW
2	Releasing duty cycle	no	0	100	75	%	RW
3	Holding duty cycle	no	0	100	11	%	RW
4	Releasing duration	no	0	65535	1000	ms	RW
5	Enable brake output	no	0	1	0	—	RW
6	Invert brake output	no	0	1	0	—	RW
7	Supply voltage	no	0	65535	240	0.1V	RW
8	Resistance	no	0	65535	440	m Ω	RW

Table 107: Entry Description ($205A_h$)

Parameter	Description
Release brake	Use this sub object to switch the brake PWM functionality 0. Brake PWM deactivated 1. Brake PWM activated
Enable brake output	Use this sub object to switch the brake functionality 0. Brake deactivated 1. Brake activated

Table 108: Parameter Description ($205A_h$)

7.1.23 Object $205F_h$: Brake Chopper

This object deals with setting up and using the brake chopper.

Object Description			
Index	Name	Object Type	Data Type
$205F_h$	Brake chopper	Variable	Record

Table 109: Object Description ($205F_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Enable	no	0	1	0	—	RW
2	Voltage limit	no	50	1000	300	0.1V	RW
3	Hysterisis	no	0	50	5	0.1V	RW
4	Type	no	0	1	0	—	RW
5	Active	no	0	1	0	—	RW
6	Supply voltage	no	0	65535	240	0.1V	RW
7	Resistance	no	0	65535	22000	mΩ	RW

Table 110: Entry Description ($205F_h$)

Parameter	Description
Type	Select the brake chopper type 0. PWM braking 1. Shunt braking

Table 111: Parameter Description ($205F_h$)

7.1.24 Object 2064_h: Reference Switch

Object Description			
Index	Name	Object Type	Data Type
2064_h	Reference switch	Variable	Record

Table 112: Object Description (2064_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Enable	no	0	3	0	—	RW
2	Polarity	no	0	3	0	—	RW

Table 113: Entry Description (2064_h)

7.1.25 Object 2069_h: Monitoring

Object Description			
Index	Name	Object Type	Data Type
2069_h	Monitoring	Variable	Record

Table 114: Object Description (2069_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Status flags	no	0	4294967295	0	—	RO
2	Supply voltage	no	0	1000	0	0.1V	RO
3	Supply voltage threshold	no	0	1000	480	0.1V	RW
4	Driver temperature	no	-20	150	0	° C	RO
5	Driver temperature threshold	no	-20	150	0	° C	RW

Table 115: Entry Description (2069_h)

7.1.26 Object $206B_h$: Lower Velocity PI

Object Description			
Index	Name	Object Type	Data Type
$206B_h$	Lower velocity PI	Variable	Record

Table 116: Object Description ($206B_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Switchover velocity	no	0	32767	0	rpm	RW
2	Lower velocity P	no	0	32767	600	—	RW
3	Lower velocity I	no	0	32767	300	—	RW

Table 117: Entry Description ($206B_h$)

7.1.27 Object 206C$_h$: Linear Drive Settings

Object Description			
Index	Name	Object Type	Data Type
206C$_h$	Linear drive settings	Variable	Record

Table 118: Object Description (206C$_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Ramp velocity	no	-2147483647	2147483647	0	μm/s	RO
2	Ramp position	no	-2147483647	2147483647	0	μm	RW
3	Velocity window	no	0	10000000	125000	μm/s	RW
4	Position window	no	0	4294967295	500000	μm	RW
5	Catchup velocity	no	0	10000000	0	μm/s	RW

Table 119: Entry Description (206C$_h$)

7.1.28 Object 206D$_h$: Driver Status

This object deals with the error handling for the driver.

Object Description			
Index	Name	Object Type	Data Type
206D$_h$	Driver status	Variable	Record

Table 120: Object Description (206D$_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	Driver status flag	no	0	65535	0	—	RO
2	Clear driver error flag	no	0	1	0	—	RW

Table 121: Entry Description (206D$_h$)

7.1.29 Object $206E_h$: Controller Initialize

Object Description			
Index	Name	Object Type	Data Type
$206E_h$	Controller initialize	Variable	Record

Table 122: Object Description ($206E_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
0	Controller initialize	no	0	1	0	–	RW

Table 123: Entry Description ($206E_h$)

7.1.30 Object $207D_h$: Switch Parameters

Object Description			
Index	Name	Object Type	Data Type
$207D_h$	Switch parameters	Variable	Record

Table 124: Object Description ($207D_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
0	Switch parameters	no	0	4294967295	0	–	RW

Table 125: Entry Description ($207D_h$)

7.1.31 Object 2082_h: Home Offset Display

Object Description			
Index	Name	Object Type	Data Type
2082_h	Home offset display	Variable	INTEGER32

Table 126: Object Description (2082_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
0	Home offset display	no	-2147483648	2147483647	0	–	RW

Table 127: Entry Description (2082_h)

7.1.32 Object 2701_h: Device Digital IO Direction

This object sets the direction for the general purpose IOs (GPIO2 to GPIO7) as either input or output.

Object Description			
Index	Name	Object Type	Data Type
2701_h	Device digital IO direction	Variable	UNSIGNED32

Table 128: Object Description (2701_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
0	Device digital IO direction	no	0	252	0	–	RW

Table 129: Entry Description (2701_h)

7.1.33 Object 2702_h: Device Digital Inputs

Bit Definitions	
Bit	Description
0	REF_R
1	REF_L
2	GPIO2
3	GPIO3
4	GPIO4
5	GPIO5
6	GPIO6
7	GPIO7

Table 130: Bit Definitions (2702_h)

Object Description			
Index	Name	Object Type	Data Type
2702_h	Device digital inputs	Variable	UNSIGNED32

Table 131: Object Description (2702_h)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
0	Device digital inputs	no	0	255	0	–	R

Table 132: Entry Description (2702_h)

7.1.34 Object 2703_h: Device Digital Outputs

With this object, the digital outputs (general purpose outputs) can be set or reset. Bits 2…7 of sub-index 1 switch the outputs of the module. Bits 2…7 of sub-index 2 determine which outputs can be switched. The number of available digital outputs depends on the module type.

Bit Definitions	
Bit	Description
2	GPO2
3	GPO3
4	GPO4
5	GPO5
6	GPO6
7	GPO7

Table 133: Bit Definitions (2703_h)

Object Description			
Index	Name	Object Type	Data Type
2703_h	Device digital outputs	Variable	ARRAY

Table 134: Object Description (2703_h)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
1	Physical outputs	rw	yes	UNSIGNED32	0
2	Output mask	rw	yes	UNSIGNED32	0

Table 135: Entry Description (2703_h)

7.1.35 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 Object 1010_h: Store Parameters sub-index 2. 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 136: 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 137: Entry Description (2704_h)

7.1.36 Object 2705_h: Node ID

The node ID can be selected using this object. 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 Object 1010_h: Store Parameters sub-index 2. 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 138: 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 139: Entry Description (2705_h)

7.1.37 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 140: 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 141: Entry Description (2707_h)

7.1.38 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 142: 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 143: Entry Description (2708_h)

7.1.39 Object $270B_h$: HAL Version

Object Description			
Index	Name	Object Type	Data Type
$270B_h$	HAL version	Variable	Record

Table 144: Object Description ($270B_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
0	HAL version	no	0	4294967295	0	–	R

Table 145: Entry Description ($270B_h$)

7.1.40 Object $270E_h$: Device Analog Inputs

Object Description			
Index	Name	Object Type	Data Type
$270E_h$	Device analog inputs	Variable	Record

Table 146: Object Description ($270E_h$)

Entry Description							
Sub-index	Name	PDO Mapping	Min	Max	Default	Unit	Access
1	ADC_u	no	0	4095	0		R
2	ADC_v	no	0	4095	0		R
3	ADC_w	no	0	4095	0		R
4	ADC voltage	no	0	4095	0		R
5	ADC temperature	no	0	4095	0		R
6	ADC ain	no	0	4095	0		R

Table 147: Entry Description ($270E_h$)

ADC value of the analog input pins AIN0 to AIN2, as well as ADC values of some on-board voltages.

8 Profile Specific Area

The profile segment contains CiA-402 standard motion control objects. These objects control the motion control functions of the TMCM-1690. Since it is not possible to operate the modes in parallel, it is possible to activate the required function by selecting a mode of operation. The control device writes to the modes of operation object 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 setpoints are used mode-specific. This implies the responsibility of the control device to avoid inconsistencies and erroneous behavior.

The following operating modes (selectable through object 6060_h, see 8.1.6) are implemented on the TMCM-1690:

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

8.1 Detailed Object Specifications

8.1.1 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:

Value Definition	
Value	Definition
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*

Table 148: Value Description ($605A_h$)

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

Table 149: Object Description ($605A_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	no	1/2/5/6	2

Table 150: Entry Description ($605A_h$)

8.1.2 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.

Value Definition	
Value	Definition
0	Disable drive function (switch off the power stage)

Table 151: Value Description ($605B_h$)

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

Table 152: Object Description ($605B_h$)

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

Table 153: Entry Description ($605B_h$)

8.1.3 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.

Value Definition	
Value	Definition
1	Slow down on slow down ramp

Table 154: Value Description ($605C_h$)

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

Table 155: Object Description ($605C_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	no	1	1

Table 156: Entry Description ($605C_h$)

8.1.4 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.

Value Definition	
Value	Definition
1	Slow down on slow down ramp and stay in *operation enabled*

Table 157: Value Description ($605D_h$)

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

Table 158: Object Description ($605D_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	no	1	1

Table 159: Entry Description ($605D_h$)

8.1.5 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.

Value Definition	
Value	Definition
2	Slow down on quick stop ramp

Table 160: Value Description ($605E_h$)

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

Table 161: Object Description ($605E_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	no	2	2

Table 162: Entry Description ($605E_h$)

8.1.6 Object 6060_h: Modes of Operation

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

Value Definition	
Value	Mode
0	No mode
1	Profile position mode (pp)
3	Profile velocity mode (pv)
6	Homing mode (hm)
8	Cyclic synchronous position mode (csp)
9	Cyclic synchronous velocity mode (csv)
10	Cyclic synchronous torque mode (cst)

Table 163: Value Description (6060_h)

The motor does not run when the operating mode is set to 0. It is 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 164: Object Description (6060_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See Table 163	0

Table 165: Entry Description (6060_h)

8.1.7 Object 6061_h: Modes of Operation Display

This object shows the operating mode that is currently set.

Value Definition	
Value	Mode
0	No mode
1	Profile position mode (pp)
3	Profile velocity mode (pv)
6	Homing mode (hm)
8	Cyclic synchronous position mode (csp)
9	Cyclic synchronous velocity mode (csv)
10	Cyclic synchronous torque mode (cst)

Table 166: Value Description (6061_h)

The motor does not run when the operating mode is set to 0. It is 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
6061_h	Modes of Operation Display	Variable	SIGNED8

Table 167: Object Description (6061_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See Table 166	0

Table 168: Entry Description (6061_h)

8.1.8 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 on the module used.

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

Table 169: Object Description ($60FD_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	UNSIGNED32	0

Table 170: Entry Description ($60FD_h$)

8.1.9 Object 6502_h: Supported Drive Modes

This object provides information on the supported drive modes. A bit that is set means the mode is supported, a bit that is not set means the mode is not supported by the drive.

Value Definition	
Bit	Mode
0	Profile position mode (pp)
1	Velocity mode (vl)
2	Profile velocity mode (pv)
3	Torque mode (tq)
4	Reserved
5	Homing mode (hm)
6	Interpolated position mode (ip)
7	Cyclic synchronous position mode (csp)
8	Cyclic synchronous velocity mode (csv)
9	Cyclic synchronous torque mode (cst)

Table 171: Value Definition (6502_h)

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

Table 172: Object Description (6502_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	UNSIGNED32	Depends on supported modes.

Table 173: Entry Description (6502_h)

8.1.10 Object $67FF_h$: Single Device Type

This object provides information on the device profile used for the individual axis. Its structure is similar to object 1000_h. The lower sixteen bits contain the device profile number, which is always 402 (0192_h) with ADI Trinamic motion control modules. The upper sixteen bits contain more information about the drive profile.

Value Definition	
Bit	Meaning
0…15	Device profile number
16	Frequency converter (0 = no, 1 = yes)
17	Servo drive (0 = no, 1 = yes)
18	Stepper motor (0 = no, 1 = yes)
19…21	Reserved
22	PDO set for generic drive device (0 = yes, 1 = no)
23	Multi device module (0 = no, 1 = yes)
24…31	Reserved

Table 174: Value Definition ($67FF_h$)

Object Description			
Index	Name	Object Type	Data Type
$67ff_h$	Single Device Type	Variable	UNSIGNED32

Table 175: Object Description ($67FF_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	UNSIGNED32	Depends on individual axis.

Table 176: Entry Description ($67FF_h$)

9 Profile Position Mode

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

See object 6060_h (section 8.1.6) on how to choose an operation mode. Object 6061_h (section 8.1.7) shows the operation mode that is set.

9.1 Detailed Object Specifications

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

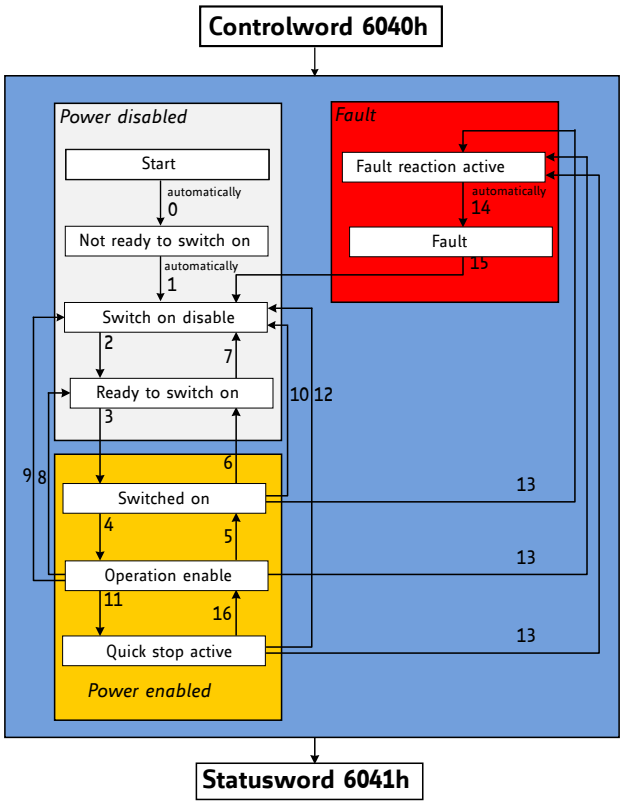


Figure 14: DS402 Finite State Machine

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.

9.1.1 Object 6040_h: Controlword

This object indicates the received command controlling the power drive system finite state machine (PDS FSM). The CiA-402 state machine can be controlled using this object. See Figure 14 for detailed information.

Structure of the Controlword											
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 177: Structure of the Controlword in pp Mode

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

Table 178: Operation Mode Specific Bits in pp Mode

Command Coding						
Command	Bits of Controlword					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 and 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 179: Command Coding

Object Description			
Index	Name	Object Type	Data Type
6040_h	Controlword	Variable	UNSIGNED16

Table 180: Object Description (6040_h in pp Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See command coding above.	

Table 181: Entry Description (6040_h in pp Mode)

9.1.2 Object 6041_h: Statusword

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. See Figure 14 for detailed information. The object is structured as defined below.

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

Structure of the Statusword															
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 182: Structure of the Statusword in pp Mode

Manufacturer Specific Bits		
Bit	Name	Definition
14	Motor activity	0: Motor stands still. 1: Motor rotates.
15	Direction of rotation	This bit shows the direction of rotation.

Table 183: Manufacturer 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. 1: Set point still in process.
13	Following error	Not supported.

Table 184: Operation Mode Specific Bits in pp Mode

State Coding	
Statusword	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 185: State Coding

Object Description			
Index	Name	Object Type	Data Type
6041_h	Controlword	Variable	UNSIGNED16

Table 186: Object Description (6041_h in pp Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See state coding above.	

Table 187: Entry Description (6041_h in pp Mode)

9.1.3 Object 6062_h: Position Demand Value

This object provides the demanded position value. The value is given in encoder steps. Object 6062_h indicates the actual position of the motor. Do not confuse it with objects 6063_h and 6064_h.

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

Table 188: Object Description (6062_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 189: Entry Description (6062_h)

9.1.4 Object 6063_h: Position Actual Internal Value

This object provides the demanded position value. The value is given in encoder steps. It is the same as object 6062_h.

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

Table 190: Object Description (6063_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 191: Entry Description (6063_h)

9.1.5 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 6063_h.

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

Table 192: Object Description (6064_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 193: Entry Description (6064_h)

9.1.6 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 reached. The value is given in increments. If the value of the position window is $FFFFFFFF_h$, the position window control is switched off. If this object is set to zero, the target reached event is signaled when the demand position (6062_h) has reached the target position (6064_h). When the position window is set to a value greater than zero, the target reached event is signaled when the actual encoder position value (6064_h) 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 194: Object Description (6067_h)

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

Table 195: Entry Description (6067_h)

9.1.7 Object 6068_h: Position Window Time

This object indicates the configured time, during which the actual position within the position window is measured. The value is given in ms. If this object is set to a value greater than zero and also the position window (6067_h) is set to a value greater than zero, the target reached event is not signaled until the actual position (6064_h) is at least as many milliseconds within the position window as defined by this object.

Object Description			
Index	Name	Object Type	Data Type
6068_h	Position Window Time	Variable	UNSIGNED16

Table 196: Object Description (6068_h)

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

Table 197: Entry Description (6068_h)

9.1.8 Object $606C_h$: Velocity Actual Value

This object shows the actual velocity value of the motor. The value is given in units of rpm.

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

Table 198: Object Description ($606C_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 199: Entry Description ($606C_h$)

9.1.9 Object $607A_h$: Target Position

The target position is the position the drive should move to in profile position mode using the current settings of motion control parameters (such as velocity, acceleration, deceleration, motion profile type, etc.). The value of this object is interpreted as absolute or relative depending on the ABS/REL flag in the controlword. It is given in encoder steps.

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

Table 200: Object Description ($607A_h$ in pp Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	SIGNED32	0

Table 201: Entry Description ($607A_h$ in pp Mode)

9.1.10 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 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 202: Object Description ($607D_h$)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
1	Minimum Position Limit	rw	no	SIGNED32	-2147483648
2	Maximum Position Limit	rw	no	SIGNED32	2147483647

Table 203: Entry Description ($607D_h$)

9.1.11 Object 6081_h: Profile Velocity

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. It is given in units of rpm.

Object Description			
Index	Name	Object Type	Data Type
6081_h	Profile Velocity	Variable	UNSIGNED32

Table 204: Object Description (6081_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	UNSIGNED32	0

Table 205: Entry Description (6081_h)

9.1.12 Object 6083_h: Profile Acceleration

This object indicates the configured acceleration. Object 6083_h sets the maximum acceleration to be used in profile position and profile velocity mode.
This value is given using rpm/s units.
In profile velocity mode, this object also sets the deceleration to be used (the deceleration ramp is always the same as the acceleration ramp in pv mode).

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

Table 206: Object Description (6083_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	UNSIGNED32	0

Table 207: Entry Description (6083_h)

9.1.13 Object 6084_h: Profile Deceleration

This object indicates the configured deceleration. Object 6084_h sets the maximum deceleration to be used in profile positioning mode.
This value is given in units of rpm/s.

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

Table 208: Object Description (6084_h)

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

Table 209: Entry Description (6084_h)

9.1.14 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 $605A_h$ is set to 2 (or 6). The value is given in the same unit as profile acceleration object 6083_h.

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

Table 210: Object Description (6085_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	no	UNSIGNED32	51200

Table 211: Entry Description (6085_h)

9.2 How to Move a Motor in pp Mode

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

- If there are no 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 (example: 500000) to object $607A_h$.
- Mark the new target position 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).

10 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 through trajectory generator.
- Monitoring of the profile velocity using a window-function.
- Monitoring of the 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

10.0.1 Object 6040_h: Controlword

This object indicates the received command controlling the power drive system finite state machine (PDS FSM). The CiA-402 state machine can be controlled using this object. See Figure 14 for detailed information.

In pv mode, the Controlword does not contain any operation mode specific bits.

Structure of the Controlword											
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 212: Structure of the Controlword in pv Mode

Command Coding						
Command	Bits of Controlword					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 and 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 213: Command Coding

Object Description			
Index	Name	Object Type	Data Type
6040_h	Controlword	Variable	UNSIGNED16

Table 214: Object Description (6040_h in pv Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See command coding above.	

Table 215: Entry Description (6040_h in pv Mode)

10.0.2 Object 6041_h: Statusword

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. See Figure 14 for detailed information. The object is structured as defined below.

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

Structure of the Statusword															
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 216: Structure of the Statusword in pv Mode

Manufacturer Specific Bits		
Bit	Name	Definition
14	Motor activity	0: Motor stands still. 1: Motor rotates.
15	Direction of rotation	This bit shows the direction of rotation.

Table 217: Manufacturer Specific Bits

Operation Mode Specific Bits in pv Mode		
Bit	Name	Definition
10	Target reached	Indicates the target speed is reached.
12	Speed	Not supported.
13	Max. slippage error	Not supported.

Table 218: Operation Mode Specific Bits in pv Mode

State Coding	
Statusword	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 219: State Coding

Object Description			
Index	Name	Object Type	Data Type
6041_h	Controlword	Variable	UNSIGNED16

Table 220: Object Description (6041_h in pv Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See state coding above	

Table 221: Entry Description (6041_h in pv Mode)

10.0.3 Object 6062_h: Position Demand Value

This object provides the demanded position value. The value is given in encoder steps. Object 6062_h indicates the actual position of the motor. Do not confuse it with objects 6063_h and 6064_h.

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

Table 222: Object Description (6062_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 223: Entry Description (6062_h)

10.0.4 Object 6063_h: Position Actual Internal Value

This object provides the demanded position value. The value is given in encoder steps. It is the same as object 6062_h.

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

Table 224: Object Description (6063_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 225: Entry Description (6063_h)

10.0.5 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 6063_h.

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

Table 226: Object Description (6064_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 227: Entry Description (6064_h)

10.0.6 Object $606C_h$: Velocity Actual Value

This object shows the actual velocity value of the motor. The value is given in units of rpm.

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

Table 228: Object Description ($606C_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 229: Entry Description ($606C_h$)

10.0.7 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 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 230: Object Description ($607D_h$)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
1	Minimum Position Limit	rw	no	SIGNED32	-2147483648
2	Maximum Position Limit	rw	no	SIGNED32	2147483647

Table 231: Entry Description ($607D_h$)

10.0.8 Object 6083_h: Profile Acceleration

This object indicates the configured acceleration. Object 6083_h sets the maximum acceleration to be used in profile position and profile velocity mode.
This value is given using rpm/s units.
In profile velocity mode, this object also sets the deceleration to be used (the deceleration ramp is always the same as the acceleration ramp in pv mode).

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

Table 232: Object Description (6083_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	UNSIGNED32	0

Table 233: Entry Description (6083_h)

10.0.9 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 $605A_h$ is set to 2 (or 6). The value is given in the same unit as profile acceleration object 6083_h.

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

Table 234: Object Description (6085_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	no	UNSIGNED32	51200

Table 235: Entry Description (6085_h)

10.0.10 Object $60FF_h$: Target Velocity

This object indicates the configured target velocity and is used as input for the trajectory generator. Object $60FF_h$ 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 6083_h and 6084_h. The values are given in rpm units.

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

Table 236: Object Description ($60FF_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	SIGNED32	0

Table 237: Entry Description ($60FF_h$)

10.1 How to Move a Motor in pv Mode

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

- If there are no limit switches connected, first disable the limit switch inputs by writing 3 to object 2005_h.
- Select pv mode by writing 3 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 speed (example: 100000) to object $60FF_h$. The motor now accelerates to that speed.
- Stop the motor by writing 0 to object $60FF_h$.

11 Homing Mode

This section 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. It is possible to specify the speeds, acceleration, and the method of homing.

There is no output data except for those bits in the statusword that 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 15 shows the defined input objects as well as the output objects. It is possible to specify the speeds, acceleration, and method of homing. The home offset object $607C_h$ allows to displace zero in the coordinate system from the home position.

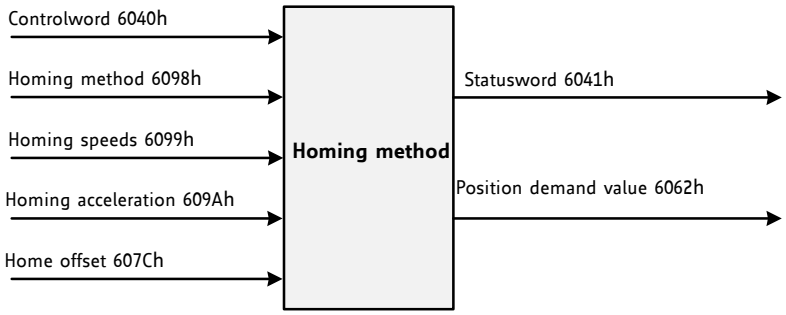


Figure 15: Homing Mode Function

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 11.2.4).

Depending on the module, there are different sources of homing methods:

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

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

11.1 Homing Methods

The TMCM-1690 supports a subset of different standard CANopen homing methods. The homing method can be chosen through object 6098_h (section 11.2.5).

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.
19	Search the positive edge of the home switch.
21	Search the negative edge of the home 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 does not move.

Table 238: Supported CANopen Homing Methods

When using homing methods that need end switch inputs or home switch inputs, take care of their configuration (object 2005_h, section 7.1.3).

11.1.1 Homing Method 17: Homing on Negative Limit Switch

Using this method, the initial direction of movement shall be leftward if the negative limit switch is inactive (here: low). The home position shall be at the point where the negative limit switch becomes inactive.

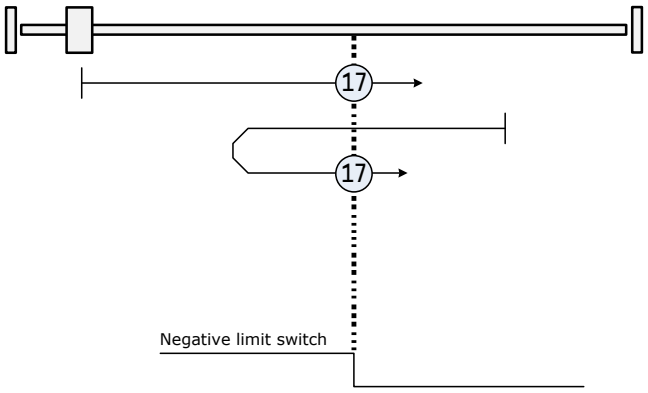


Figure 16: Homing Method 17

11.1.2 Homing Method 18: Homing on Positive Limit Switch

Using this method, the initial direction of movement shall be rightward if the positive limit switch is inactive (here: low). The home position shall be at the point where the positive limit switch becomes inactive.

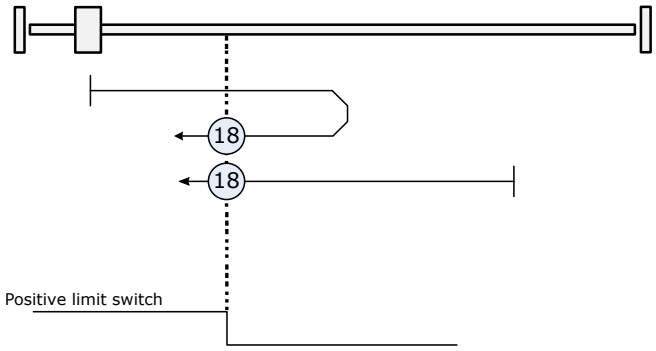


Figure 17: Homing Method 18

11.1.3 Homing Method 19: Homing on Positive Home Switch

Using this method, the initial direction of movement shall be dependent on the state of the home switch. The home position shall be at the point where the home switch changes state. If the initial direction of movement leads away from the home switch, the drive shall reverse on encountering the relevant limit switch.

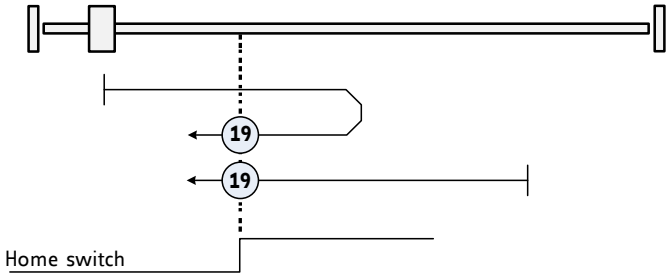


Figure 18: Homing Method 19

11.1.4 Homing Method 21: Homing on Negative Home Switch

Using this method, the initial direction of movement shall be dependent on the state of the home switch. The home position shall be at the point where the home switch changes state. If the initial direction of movement leads away from the home switch, the drive shall reverse on encountering the relevant limit switch.

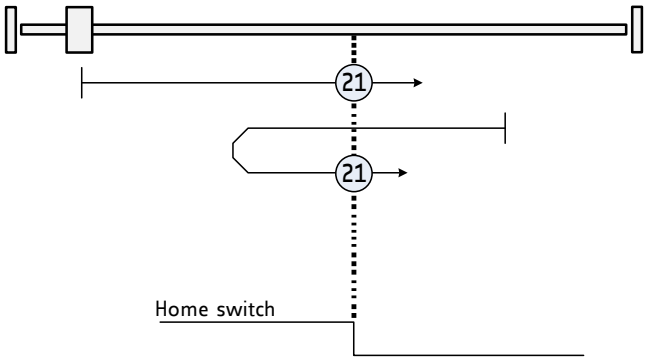


Figure 19: Homing Method 21

11.1.5 Homing Method 35: Current Position as Home Position

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

11.2 Detailed Object Specifications

11.2.1 Object 6040_h: Controlword

This object indicates the received command controlling the power drive system finite state machine (PDS FSM). The CiA-402 state machine can be controlled using this object. See Figure 14 for detailed information.

Structure of the Controlword										
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 239: Structure of the Controlword 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 240: Operation Mode Specific Bits in hm Mode

Command Coding						
Command	Bits of Controlword					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 and 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 241: Command Coding

Object Description			
Index	Name	Object Type	Data Type
6040_h	Controlword	Variable	UNSIGNED16

Table 242: Object Description (6040_h in hm Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See command coding above.	

Table 243: Entry Description (6040_h in hm Mode)

11.2.2 Object 6041_h: Statusword

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. See Figure 14 for detailed information. The object is structured as defined below.

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

Structure of the Statusword															
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; s o =switch on.

Table 244: Structure of the Statusword in hm Mode

Manufacturer Specific Bits		
Bit	Name	Definition
14	Motor activity	0: Motor stands still. 1: Motor rotates.
15	Direction of rotation	This bit shows the direction of rotation.

Table 245: Manufacturer Specific Bits

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

Table 246: Operation Mode Specific Bits in hm Mode

State Coding	
Statusword	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 247: State Coding

Object Description			
Index	Name	Object Type	Data Type
6041_h	Controlword	Variable	UNSIGNED16

Table 248: Object Description (6041_h in hm Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See state coding above.	

Table 249: Entry Description (6041_h in hm Mode)

11.2.3 Object $606C_h$: Velocity Actual Value

This object shows the actual velocity value of the motor. The value is given in units of rpm.

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

Table 250: Object Description ($606C_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 251: Entry Description ($606C_h$)

11.2.4 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. The value of this object is given in encoder steps. Negative values indicate the opposite direction.

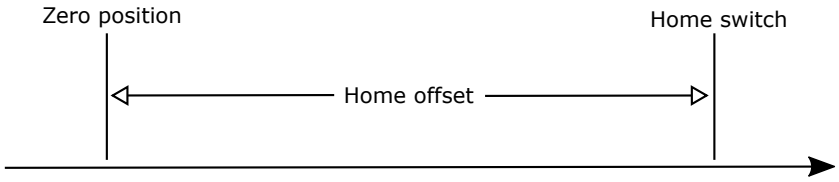


Figure 20: Home Offset

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

Table 252: Object Description ($607C_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	no	SIGNED32	0

Table 253: Entry Description ($607C_h$)

11.2.5 Object 6098_h: Homing Method

The homing method to be used can be selected by writing to this object. See Table 238 for a list of homing methods supported by the current version of the TMCM-1690 CANopen firmware.

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

Table 254: Object Description (6098_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	no	SIGNED8	0

Table 255: Entry Description (6098_h)

11.2.6 Object 6099_h: Homing Speeds

This object indicates the configured speeds used during homing procedure. The values are given in rpm units. Using object 6099_h, a fast and a slow homing speed can be set. In most homing modes, the home switch is searched with the fast speed first. When the home switch is found, the motor is decelerated to the slow speed (using the homing acceleration, object $609A_h$) to search for the exact switch point. When the switch point is found, the motor is stopped at that point.

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

Table 256: Object Description (6099_h)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
1	Fast homing speed	rw	no	UNSIGNED32	0
2	Slow homing speed	rw	no	UNSIGNED32	0

Table 257: Entry Description (6099_h)

11.2.7 Object $609A_h$: Homing Acceleration

This object indicates the configured acceleration and deceleration to be used during homing operation. This object uses rpm/s units.

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

Table 258: Object Description ($609A_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	no	UNSIGNED32	0

Table 259: Entry Description ($609A_h$)

11.3 How to Start a Homing in hm Mode

Here is a small example that shows how to home the motor in hm mode. In this example, assume the module is 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 19 by writing 19 to object 6098_h.
- Set the homing speeds by writing, example, 50000 to object 6099_h subindex 1 and, example, 10000 to object 6099_h subindex 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.

12 Cyclic Synchronous Position Mode

The cyclic synchronous position mode is used to directly control the position of the motor. It contains limit functions, but not a trajectory generator. The trajectory generator is located in the control device (the manager), not in the drive device. In cyclic synchronous manner, the control device provides a target position to the drive device, which performs position control, velocity control, and torque control.

The main control parameters are the target position (object $607A_h$, see section 12.1.7) and the interpolation time period (object $60C2_h$, see section 12.1.10). The drive automatically sets the velocity in such a manner that the next target position is reached within the interpolation time period. Acceleration and deceleration ramps are not used in this mode.

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

- Position demand value input directly through an object.
- Monitoring of the position.
- Limiting the position using the software limits or the hardware limit switches.

12.1 Detailed Object Specifications

12.1.1 Object 6040_h: Controlword

This object indicates the received command controlling the power drive system finite state machine (PDS FSM). The CiA-402 state machine can be controlled using this object. See Figure 14 for detailed information. The cyclic synchronous position mode does not use any mode specific bits of the Controlword.

Structure of the Controlword									
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 260: Structure of the Controlword in csp Mode

Command Coding						
Command	Bits of Controlword					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 and 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 261: Command Coding

Object Description			
Index	Name	Object Type	Data Type
6040_h	Controlword	Variable	UNSIGNED16

Table 262: Object Description (6040_h in csp Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See command coding above.	

Table 263: Entry Description (6040_h in csp Mode)

12.1.2 Object 6041_h: Statusword

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. See Figure 14 for detailed information. The object is structured as defined below.
For more information about the coding, refer to the CANopen drives and motion control device profile, part 2.

Structure of the Statusword															
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 264: Structure of the Statusword in csp Mode

Manufacturer Specific Bits		
Bit	Name	Definition
14	Motor activity	0: Motor stands still. 1: Motor rotates.
15	Direction of rotation	This bit shows the direction of rotation.

Table 265: Manufacturer Specific Bits

Operation Mode Specific Bits in csp Mode		
Bit	Name	Definition
10	Reserved	Not used.
12	Target position ignored	0: Target position ignored. 1: Target position used as input to position controller.
13	Following error	0: No following error. 1: Following error.

Table 266: Operation Mode Specific Bits in csp Mode

State Coding	
Statusword	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 267: State Coding

Object Description			
Index	Name	Object Type	Data Type
6041_h	Controlword	Variable	UNSIGNED16

Table 268: Object Description (6041_h in csp Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See state coding above	

Table 269: Entry Description (6041_h in csp Mode)

12.1.3 Object 6062_h: Position Demand Value

This object provides the demanded position value. The value is given in encoder steps. Object 6062_h indicates the actual position of the motor. Do not confuse it with objects 6063_h and 6064_h.

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

Table 270: Object Description (6062_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 271: Entry Description (6062_h)

12.1.4 Object 6063_h: Position Actual Internal Value

This object provides the demanded position value. The value is given in encoder steps. It is the same as object 6062_h.

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

Table 272: Object Description (6063_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 273: Entry Description (6063_h)

12.1.5 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 6063_h.

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

Table 274: Object Description (6064_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 275: Entry Description (6064_h)

12.1.6 Object $606C_h$: Velocity Actual Value

This object shows the actual velocity value of the motor. The value is given in units of rpm.

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

Table 276: Object Description ($606C_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 277: Entry Description ($606C_h$)

12.1.7 Object $607A_h$: Target Position

The target position is the position the drive should move to in cyclic synchronous position mode using the current interpolation time period. In csp mode, this value is always interpreted as an absolute value.

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

Table 278: Object Description ($607A_h$ in csp Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	SIGNED32	0

Table 279: Entry Description ($607A_h$ in csp Mode)

12.1.8 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 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 280: Object Description ($607D_h$)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
1	Minimum Position Limit	rw	no	SIGNED32	-2147483648
2	Maximum Position Limit	rw	no	SIGNED32	2147483647

Table 281: Entry Description ($607D_h$)

12.1.9 Object $60B0_h$: Position Offset

This object provides an offset to the target position (object $607A_h$, see section 12.1.7). The value is given in encoder steps and is added to the target position.

Object Description			
Index	Name	Object Type	Data Type
$60B0_h$	Position Offset	Variable	SIGNED32

Table 282: Object Description ($60B0_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	-2147483648...2147483647	0

Table 283: Entry Description ($60B0_h$)

12.1.10 Object $60C2_h$: Interpolation Time Period

This object indicates the interpolation cycle time. The interpolation time period (subindex 01_h) is given in $10^{interpolation\_time\_index}$ s. The interpolation time index (subindex 02_h) is dimensionless.

Object Description			
Index	Name	Object Type	Data Type
$60C2_h$	Interpolation Time Period	Record	Interpolation time period record (0080_h)

Table 284: Object Description ($60C2_h$)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
0	Highest sub-index supported	ro	no	UNSIGNED8	2
1	Interpolation Time Period Value	rw	no	UNSIGNED8	1
2	Interpolation Time Index	rw	no	-3...3	-3

Table 285: Entry Description ($60C2_h$)

13 Cyclic Synchronous Velocity Mode

The cyclic synchronous velocity mode is used to directly control the velocity of the motor. It contains limit functions, but not a trajectory generator. The trajectory generator is located in the control device (the manager), not in the drive device. In cyclic synchronous manner, the control device provides a target velocity to the drive device, which performs position control, velocity control, and torque control.

The main control parameters are the target velocity (object $60FF_h$, see section 13.1.4) and the interpolation time period (object $60C2_h$, see section 13.1.7). The drive automatically sets the acceleration in such a manner that the next target velocity is reached within the interpolation time period. Acceleration and deceleration ramps are not used in this mode.

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

- Velocity demand value input directly through an object.
- Monitoring of the position.
- Limiting the position using the software limits or hardware limit switches.

13.1 Detailed Object Specifications

13.1.1 Object 6040_h: Controlword

This object indicates the received command controlling the power drive system finite state machine (PDS FSM). The CiA-402 state machine can be controlled using this object. See Figure 14 for detailed information. The cyclic synchronous velocity mode does not use any mode specific bits of the Controlword.

Structure of the Controlword									
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 286: Structure of the Controlword in csv Mode

Command Coding						
Command	Bits of Controlword					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 and 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 287: Command Coding

Object Description			
Index	Name	Object Type	Data Type
6040_h	Controlword	Variable	UNSIGNED16

Table 288: Object Description (6040_h in csv Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See command coding above.	

Table 289: Entry Description (6040_h in csv Mode)

13.1.2 Object 6041_h: Statusword

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. See Figure 14 for detailed information. The object is structured as defined below.

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

Structure of the Statusword															
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 290: Structure of the Statusword in csv Mode

Manufacturer Specific Bits		
Bit	Name	Definition
14	Motor activity	0: Motor stands still. 1: Motor rotates.
15	Direction of rotation	This bit shows the direction of rotation.

Table 291: Manufacturer Specific Bits

Operation Mode Specific Bits in csv Mode		
Bit	Name	Definition
10	Reserved	Not used.
12	Target position ignored	0: Target velocity ignored. 1: Target velocity used as input to velocity controller.
13	Reserved	Not used.

Table 292: Operation Mode Specific Bits in csv Mode

State Coding	
Statusword	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 293: State Coding

Object Description			
Index	Name	Object Type	Data Type
6041_h	Controlword	Variable	UNSIGNED16

Table 294: Object Description (6041_h in csv Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See state coding above	

Table 295: Entry Description (6041_h in csv Mode)

13.1.3 Object $606C_h$: Velocity Actual Value

This object shows the actual velocity value of the motor. The value is given in units of rpm.

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

Table 296: Object Description ($606C_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 297: Entry Description ($606C_h$)

13.1.4 Object $60FF_h$: Target Velocity

In csv mode, the target velocity specifies the velocity to be reached within the interpolation time period. The values are given in rpm units.

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

Table 298: Object Description ($60FF_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	SIGNED32	0

Table 299: Entry Description ($60FF_h$)

13.1.5 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 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 300: Object Description ($607D_h$)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
1	Minimum Position Limit	rw	no	SIGNED32	-2147483648
2	Maximum Position Limit	rw	no	SIGNED32	2147483647

Table 301: Entry Description ($607D_h$)

13.1.6 Object $60B1_h$: Velocity Offset

This object provides an offset to the target velocity (object $60FF_h$, see section 13.1.4). The value is added to the target velocity.

Object Description			
Index	Name	Object Type	Data Type
$60B1_h$	Velocity Offset	Variable	SIGNED32

Table 302: Object Description ($60B1_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	-2147483648…2147483647	0

Table 303: Entry Description ($60B1_h$)

13.1.7 Object $60C2_h$: Interpolation Time Period

This object indicates the interpolation cycle time. The interpolation time period (subindex 01_h) is given in $10^{interpolation\_time\_index}$ s. The interpolation time index (subindex 02_h) is dimensionless.

Object Description			
Index	Name	Object Type	Data Type
$60C2_h$	Interpolation Time Period	Record	Interpolation time period record (0080_h)

Table 304: Object Description ($60C2_h$)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
0	Highest sub-index supported	ro	no	UNSIGNED8	2
1	Interpolation Time Period Value	rw	no	UNSIGNED8	1
2	Interpolation Time Index	rw	no	-3…3	-3

Table 305: Entry Description ($60C2_h$)

14 Cyclic Synchronous 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 through an object.
- Monitoring of the torque.
- Limiting the position using the software limits or hardware limit switches.

14.1 Detailed Object Specifications

14.1.1 Object 6040_h: Controlword

This object indicates the received command controlling the power drive system finite state machine (PDS FSM). The CiA-402 state machine can be controlled using this object. See Figure 14 for detailed information. The cyclic synchronous torque mode does not use any mode specific bits of the Controlword.

Structure of the Controlword									
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 306: Structure of the Controlword in cst Mode

Command Coding						
Command	Bits of Controlword					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 and 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 307: Command Coding

Object Description			
Index	Name	Object Type	Data Type
6040_h	Controlword	Variable	UNSIGNED16

Table 308: Object Description (6040_h in cst Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See command coding above.	

Table 309: Entry Description (6040_h in cst Mode)

14.1.2 Object 6041_h: Statusword

This object provides the status of the PDS FSA. It reflects the status of the CiA-402 state machine. See Figure 14 for detailed information. The object is structured as defined below.

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

Structure of the Statusword															
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 310: Structure of the Statusword in cst Mode

Manufacturer Specific Bits		
Bit	Name	Definition
14	Motor activity	0: Motor stands still. 1: Motor rotates.
15	Direction of rotation	This bit shows the direction of rotation.

Table 311: Manufacturer Specific Bits

Operation Mode Specific Bits in cst Mode		
Bit	Name	Definition
10	Reserved	Not used.
12	Target torque ignored	0: Target torque ignored. 1: Target torque used as input to control loop.
13	Reserved	Not used.

Table 312: Operation Mode Specific Bits in cst Mode

State Coding	
Statusword	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 313: State Coding

Object Description			
Index	Name	Object Type	Data Type
6041_h	Controlword	Variable	UNSIGNED16

Table 314: Object Description (6041_h in cst Mode)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	See state coding above	

Table 315: Entry Description (6041_h in cst Mode)

14.1.3 Object 6062_h: Position Demand Value

This object provides the demanded position value. The value is given in encoder steps. Object 6062_h indicates the actual position of the motor. Do not confuse it with objects 6063_h and 6064_h.

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

Table 316: Object Description (6062_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 317: Entry Description (6062_h)

14.1.4 Object 6063_h: Position Actual Internal Value

This object provides the demanded position value. The value is given in encoder steps. It is the same as object 6062_h.

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

Table 318: Object Description (6063_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 319: Entry Description (6063_h)

14.1.5 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 6063_h.

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

Table 320: Object Description (6064_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	SIGNED32	no

Table 321: Entry Description (6064_h)

14.1.6 Object 6071_h: Target Torque

This object sets the desired torque value. The value is given in mA.

Object Description			
Index	Name	Object Type	Data Type
6071_h	Target torque	Variable	INTEGER16

Table 322: Object Description (6071_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	-32768...32767	0

Table 323: Entry Description (6071_h)

14.1.7 Object 6077_h: Torque Actual Value

This object provides the actual torque value. The value is given in mA.

Object Description			
Index	Name	Object Type	Data Type
6077_h	Torque actual value	Variable	INTEGER16

Table 324: Object Description (6077_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	yes	-32768...32767	0

Table 325: Entry Description (6077_h)

14.1.8 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 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 326: Object Description ($607D_h$)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
1	Minimum Position Limit	rw	no	SIGNED32	-2147483648
2	Maximum Position Limit	rw	no	SIGNED32	2147483647

Table 327: Entry Description ($607D_h$)

14.1.9 Object $60B2_h$: Torque Offset

This object provides an offset to the torque value. It is added to the target torque (object 6071_h, see Section 14.1.6).

Object Description			
Index	Name	Object Type	Data Type
$60B2_h$	Torque Offset	Variable	SIGNED16

Table 328: Object Description ($60B2_h$)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	rw	yes	-32768...32767	0

Table 329: Entry Description ($60B2_h$)

14.1.10 Object $60C2_h$: Interpolation Time Period

This object indicates the interpolation cycle time. The interpolation time period (subindex 01_h) is given in $10^{interpolation\_time\_index}$ s. The interpolation time index (subindex 02_h) is dimensionless.

Object Description			
Index	Name	Object Type	Data Type
$60C2_h$	Interpolation Time Period	Record	Interpolation time period record (0080_h)

Table 330: Object Description ($60C2_h$)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
0	Highest sub-index supported	ro	no	UNSIGNED8	2
1	Interpolation Time Period Value	rw	no	UNSIGNED8	1
2	Interpolation Time Index	rw	no	-3…3	-3

Table 331: Entry Description ($60C2_h$)

15 Emergency Messages

The module sends an emergency (EMCY) 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.

15.1 Format of Emergency Messages

The COB-ID of an emergency (EMCY) message is 80_h + ID of the node. So, for example, it is 81_h for node #1. The first two bytes contain the error code with LSB first. The third data byte contains the content of the error register (object 1001_h). The other five bytes can contain additional information, depending on the error code.

EMCY								
COB-ID	1	2	3	4	5	6	7	8
$0x80_h$+Node ID	Error code		Error register	Additional bytes				
	LSB	MSB	(1001_h)	#1…#5				

Table 332: Format of EMCY Messages

15.2 Error Codes Used by the TMCM-1690 Module

The following table shows all EMCY error codes used by the TMCM-1690 module.

Emergency Messages (EMCY) of the TMCM-1690						
	Additional byte					
Error code	1	2	3	4	5	Description
0000_h	0	0	0	0	0	**Fault reset** The fault reset command has been executed.
4310_h	2	0	0	0	0	**Overtemperature error** The motor driver has been switched off because the temperature limit has been exceeded.
5441_h	0	255	0	0	0	**Shutdown switch active** 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** The data in the received PDO is either wrong or cannot be accepted due to the internal state of the drive.
8100_h	0	255	0	0	0	**Communication error** General CAN bus communication error.
8110_h	1	255	0	0	0	**CAN controller overflow** The receive message buffer of the CAN controller hardware is full and some CAN messages are lost.

	Additional byte					
Error code	1	2	3	4	5	Description
8110_h	2	255	0	0	0	**CAN Tx buffer overflow** The software CAN transmit buffer is full and so some CAN messages are lost.
8110_h	3	255	0	0	0	**CAN Rx buffer overflow** The software CAN receive buffer is full and so some CAN messages are lost.
8120_h	0	255	0	0	0	**CAN error passive** The CAN controller has detected communication errors and has entered the CAN Error passive state.
8130_h	0	255	0	0	0	**Heartbeat or lifeguard error** The module did not receive a heartbeat or lifeguard message in time.
8140_h	0	255	0	0	0	**CAN controller recovered from bus-off state** 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.
8210_h	0	255	0	0	0	**PDO not processed due to length error** A PDO sent to the module cannot be processed because too few bytes are supplied.
8220_h	0	255	0	0	0	**PDO length exceeded** A PDO sent to the module cannot be processed because too many bytes are supplied.
8611_h	0	0	0	0	0	**Following error** The deviation between motor position counter and encoder position counter has exceeded the following error window.
$ff00_h$	0	0	0	0	0	**Undervoltage** The supply voltage is too low to drive a motor.
$ff01_h$	1	0	0	0	0	**Positive software limit** The actual position is outside the range defined by object $607d_h$.
$ff01_h$	2	0	0	0	0	**Negative software limit** The actual position is outside the range defined by object $607d_h$.
$ff01_h$	3	0	0	0	0	**Positive limit switch** The positive limit switch has been touched outside of the homing function.
$ff01_h$	4	0	0	0	0	**Negative limit switch** The negative limit switch has been touched outside of the homing function.

Table 333: Emergency Messages (EMCY)

16 SDO Abort Codes

Trying to access an object through SDO read or SDO write may result in an error. In such a case, an SDO abort transfer message containing an abort code is sent. The following table lists all SDO abort codes defined by the CiA-301 standard. Not all of these are used by the TMCM-1690 module.

SDO Abort Codes	
Abort Code	Description
05030000_h	Toggle bit not alternated.
05040000_h	SDO protocol timed out.
05040001_h	Client/server command specifier not valid or unknown.
05040002_h	Invalid block size.
05040003_h	Invalid sequence number.
05040004_h	CRC error.
05040005_h	Out of memory.
06010000_h	Unsupported access to an object.
06010001_h	Attempt to read a write only object.
06010002_h	Attempt to write a read only object.
06020000_h	Object does not exist in object dictionary.
06040041_h	Object cannot be mapped to the PDO.
06040042_h	The number and length of the objects to be mapped exceed the PDO length.
06040043_h	General parameter incompatibility reason.
06040047_h	General internal incompatibility in the device.
06060000_h	Access failed due to a hardware error.
06070010_h	Data type does not match, length of service parameter does not match.
06070012_h	Data type does not match, length of service parameter too high.
06070013_h	Data type does not match, length of service parameter too low.
06090011_h	Subindex does not exist.
06090030_h	Invalid value for parameter.
06090031_h	Value of parameter too high.
06090032_h	Value of parameter too low.
06090036_h	Maximum value is less than minimum value.
$060A0023_h$	Resource not available.
08000000_h	General error.
08000020_h	Data cannot be transferred to or stored in the application.
08000021_h	Data cannot be transferred to or stored in the application because of local control.

Abort Code	Description
08000022_h	Data cannot be transferred to or stored in the application because of the present device state.
08000023_h	Object dictionary dynamic generation failed or no object dictionary is present.
08000024_h	No data available.

Table 334: SDO Abort Codes

17 Firmware Update

For an update of the TMCM-1690's firmware use the TMCL-IDE.

Download the required update files from the TMCM-1690's product page.
An update over CAN is only possible through the Trinamic TMCL CAN protocol. Because of that, prior to starting the upload, the TMCM-1690 must be put into boot mode by writing the value 12345678_h into object $5FFF_h$.

Note Make sure to update only CANopen firmware files.

18 Figures Index

19 Tables Index

20 Supplemental Directives

20.1 Producer Information

20.2 Copyright

ADI Trinamic/Trinamic Motion Control GmbH & Co. KG owns the content of this user manual in its entirety, including but not limited to pictures, logos, trademarks, and resources.

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

20.3 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 ADI Trinamic or by other manufacturers, whose products are used or referred to in combination with ADI Trinamic's products and ADI 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.

20.4 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.

20.5 Disclaimer: Life Support Systems

ADI Trinamic/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 ADI Trinamic/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.

20.6 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 ADI Trinamic's/Trinamic Motion Control GmbH & Co. KG specific written consent.

ADI Trinamic/Trinamic Motion Control GmbH & Co. KG 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 ADI Trinamic/Trinamic Motion Control GmbH & Co. KG.
ADI Trinamic/Trinamic Motion Control GmbH & Co. KG conveys no patent, copyright, mask work right or other trademark right to this product. ADI Trinamic/Trinamic Motion Control GmbH & Co. KG assumes no liability for any patent and/or other trademark rights of a third party resulting from processing or handling of the product and/or any other use of the product.

20.7 Collateral Documents and 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.analog.com.

21 Revision History

21.1 Firmware Revision

Version	Date	Description
1.00	06/23	First release
1.01	07/23	New release: • Fixed on the fly change of motion parameters • Fixed issue with the block hall
1.03	10/24	Bugfix and update release • Fixed overshoot in ramp position for linear ramp. • Added support for 12 bits axis parameters. • Added support for IC MU_150 absolute encoder. • Added initialization routines for abn encoder for linear drive. • Increased resolution of abn encoder upto 1050000. • Added a catchup velocity object which is used in position mode for actual position to jump up to ramp position. • Added sub objects for linear stage velocity and position window. • Removed redundant linear drive parameters to merge them with the profile specific area parameters. • Increased linear stage acceleration and velocity maximum limits.

Table 335: Firmware Revision

21.2 Document Revision

Version	Date	Description
0	02/24	First release
1	04/25	Firmware update v1.03

Table 336: Document Revision