



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

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

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



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

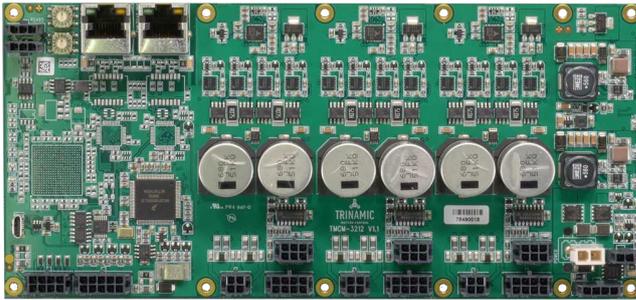
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



TMCM-3212 CANopen[®] Firmware Manual

Firmware Version V3.19 | Document Revision V1.02 • 2016-NOV-29

The TMCM-3212 is a three axes controller/driver module for 2-phase bipolar stepper motors with separate differential encoder and separate home and stop switch inputs for each axis. Dynamic current control, and quiet, smooth and efficient operation are combined with stealthChop[™], dcStep[™], stallGuard[™] and coolStep[™] features. The module offers four analog or digital inputs as well as four digital outputs in combination with a break chopper unit.



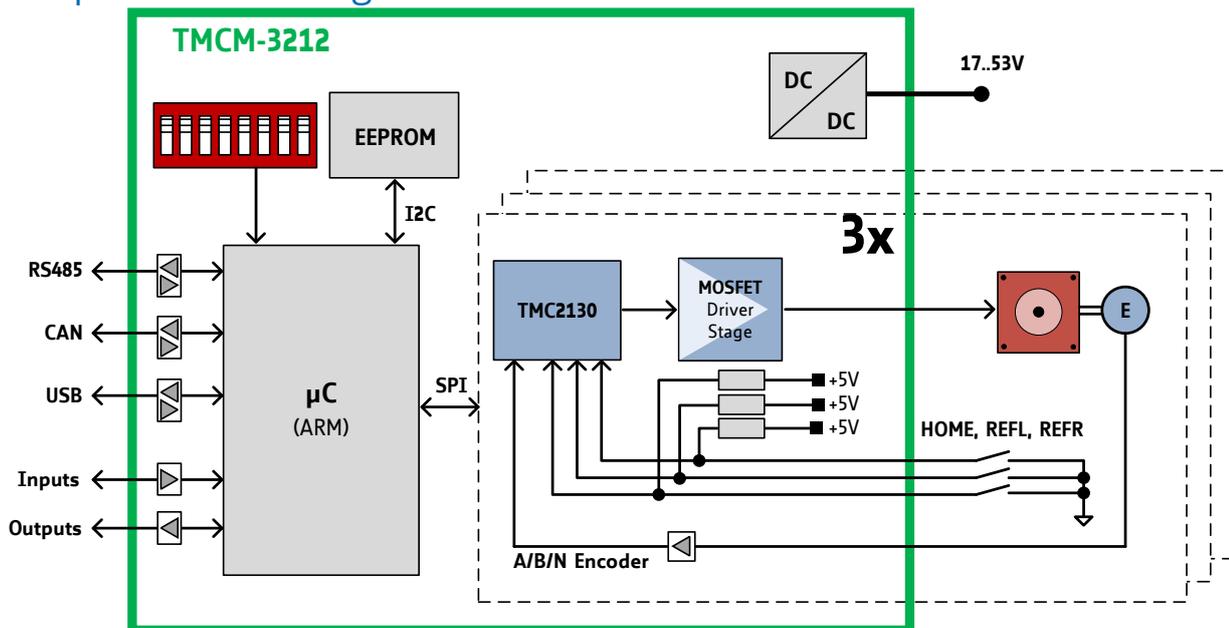
Features

- 3-Axes Stepper Motor Control
- CANopen[®] CiA-402 Drive Profile
- Encoder Support
- coolStep[™]
- dcStep[™]
- stallGuard2[™]
- stealthChop[™]

Applications

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

Simplified Block Diagram



©2016 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany
 Terms of delivery and rights to technical change reserved.
 Download newest version at: www.trinamic.com



Read entire documentation.

Contents

1	Preface	6
1.1	General Features of this CANopen Implementation	6
1.2	Abbreviations used in this Manual	7
1.3	Firmware Update	7
1.4	Trinamic's unique Features — easy to use with CANopen	8
1.4.1	stallGuard2	8
1.4.2	coolStep	8
2	Communication	10
2.1	Reference Model	10
2.2	NMT State Machine	12
2.3	Device Model	13
2.4	Object Dictionary	14
2.4.1	Object Indices on Multi-Axis Modules	15
3	Communication area	16
3.1	Detailed object specifications	16
3.1.1	Object 1000 _h : Device Type	16
3.1.2	Object 1001 _h : Error Register	16
3.1.3	Object 1005 _h : COB-ID SYNC Message	17
3.1.4	Object 1008 _h : Manufacturer Device Name	18
3.1.5	Object 1009 _h : Manufacturer Hardware Version	18
3.1.6	Object 100A _h : Manufacturer Software Version	18
3.1.7	Object 100C _h : Guard Time	19
3.1.8	Object 100D _h : Life Time Factor	19
3.1.9	Object 1010 _h : Store Parameters	19
3.1.10	Object 1011 _h : Restore Parameters	21
3.1.11	Object 1014 _h : COB-ID Emergency Object	22
3.1.12	Object 1015 _h : Inhibit Time EMCY	23
3.1.13	Object 1016 _h : Consumer Heartbeat Time	23
3.1.14	Object 1017 _h : Producer Heartbeat Time	24
3.1.15	Object 1018 _h : Identity Object	24
3.1.16	Object 1029 _h : Error Behaviour	25
3.1.17	Objects 1400 _h – 1403 _h : Receive PDO Communication Parameter	25
3.1.18	Objects 1600 _h – 1603 _h : Receive PDO Mapping Parameter	26
3.1.19	Objects 1800 _h – 1803 _h : Transmit PDO Communication Parameter	27
3.1.20	Objects 1A00 _h – 1A03 _h : Transmit PDO Mapping Parameter	29
4	Manufacturer specific area	31
4.1	Objects related to coolStep	31
4.2	Detailed object specifications	34
4.2.1	Object 2000 _h : Microstep resolution	34
4.2.2	Object 2001 _h : Fullstep resolution	34
4.2.3	Object 2002 _h : Brake delay times	34
4.2.4	Object 2003 _h : Maximum current	35
4.2.5	Object 2004 _h : Standby current	36
4.2.6	Object 2005 _h : Limit switches	36
4.2.7	Object 200A _h : Enable drive delay time	37
4.2.8	Object 200B _h : Encoder parameters	37
4.2.9	Object 200C _h : Brake current feed	38
4.2.10	Object 2010 _h : Profile Start Velocity	38
4.2.11	Object 2011 _h : Profile A1	39



4.2.12 Object 2012 _h : Profile V1	39
4.2.13 Object 2013 _h : Profile D1	40
4.2.14 Object 2015 _h : Ramp Wait Time	40
4.2.15 Object 2089 _h : Setting Delay	40
4.2.16 Object 208C _h : Velocity Dimension Index	41
4.2.17 Object 208E _h : Acceleration Dimension Index	41
4.2.18 Object 2092 _h : Chopper Blank Time	42
4.2.19 Object 2093 _h : Chopper Mode	42
4.2.20 Object 2094 _h : Chopper Hysteresis Decrement	43
4.2.21 Object 2095 _h : Chopper Hysteresis End	43
4.2.22 Object 2096 _h : Chopper Hysteresis Start	44
4.2.23 Object 2097 _h : Chopper Off Time	44
4.2.24 Object 2098 _h : Smart Energy Current Minimum	44
4.2.25 Object 2099 _h : Smart Energy Current Down Step	45
4.2.26 Object 209A _h : Smart Energy Hysteresis	45
4.2.27 Object 209B _h : Smart Energy Current Up Step	46
4.2.28 Object 209C _h : Smart Energy Hysteresis Start	46
4.2.29 Object 209D _h : Smart Energy Filter Enable	47
4.2.30 Object 209E _h : stallGuard2 Threshold	47
4.2.31 Object 20A1 _h : Short Protection Disable	48
4.2.32 Object 20A3 _h : Vsense	48
4.2.33 Object 20A4 _h : Stop on Stall	49
4.2.34 Object 20A5 _h : Smart Energy Threshold Speed	49
4.2.35 Object 20B0 _h : PWM Threshold Speed	50
4.2.36 Object 20B1 _h : PWM Gradient	50
4.2.37 Object 20B2 _h : PWM Amplitude	51
4.2.38 Object 20B3 _h : dcStep Minimum Speed	51
4.2.39 Object 20B4 _h : dcStep Time	51
4.2.40 Object 20B5 _h : dcStep stallGuard	52
4.2.41 Object 20B6 _h : Fullstep Threshold Speed	52
4.2.42 Object 20B7 _h : High Speed Chopper Mode	53
4.2.43 Object 20B8 _h : High Speed Fullstep Mode	53
4.2.44 Object 20B9 _h : Power Down Ramp	53
4.2.45 Object 2100 _h : Home Offset Display	54
4.2.46 Object 2101 _h : Actual Load Value	54
4.2.47 Object 2102 _h : Driver Error Flags	55
4.2.48 Object 2107 _h : Microstep resolution display	55
4.2.49 Object 210B _h : Step Counter	56
4.2.50 Object 2121 _h : PWM Scale Value	56
4.2.51 Object 2122 _h : Measured Velocity	57
4.2.52 Object 2700 _h : TMCL Direct Communication	57
4.2.53 Object 2701 _h : Manufacturer Specific Mode	58
4.2.54 Object 2702 _h : Device Digital Inputs	58
4.2.55 Object 2703 _h : Device Digital Outputs	59
4.2.56 Object 2704 _h : CAN Bit Rate	60
4.2.57 Object 2705 _h : Node ID	60
4.2.58 Object 2706 _h : Store	61
4.2.59 Object 2707 _h : CAN Bit Rate Load	61
4.2.60 Object 2708 _h : Node ID Load	61
4.2.61 Object 270E _h : Device Analog Inputs	62



5	Profile specific area	63
5.1	Detailed object specifications	63
5.1.1	Object 605A _h : Quick stop option code	63
5.1.2	Object 605B _h : Shutdown option code	64
5.1.3	Object 605C _h : Disable operation option code	65
5.1.4	Object 605D _h : Halt option code	65
5.1.5	Object 605E _h : Fault reaction option code	66
5.1.6	Object 6060 _h : Modes of operation	66
5.1.7	Object 6061 _h : Modes of operation	67
5.1.8	Object 606A _h : Sensor selection code	68
5.1.9	Object 608F _h : Position Encoder Resolution	68
5.1.10	Object 60FD _h : Digital Inputs	69
5.1.11	Object 6502 _h : Supported Drive Modes	69
6	Profile position mode	71
6.1	Detailed Object Specifications	71
6.1.1	Object 6040 _h : Control Word	72
6.1.2	Object 6041 _h : Status Word	73
6.1.3	Object 6062 _h : Position Demand Value	74
6.1.4	Object 6063 _h : Position Actual Internal Value	75
6.1.5	Object 6064 _h : Position Actual Value	75
6.1.6	Object 6065 _h : Following Error Window	76
6.1.7	Object 6067 _h : Position Window	76
6.1.8	Object 6068 _h : Position Window Time	77
6.1.9	Object 606C _h : Velocity Actual Value	77
6.1.10	Object 607A _h : Target Position	78
6.1.11	Object 607D _h : Software Position Limit	78
6.1.12	Object 6081 _h : Profile Velocity	79
6.1.13	Object 6082 _h : End Velocity	79
6.1.14	Object 6083 _h : Profile Acceleration	80
6.1.15	Object 6084 _h : Profile Deceleration	80
6.1.16	Object 6085 _h : Quick Stop Deceleration	80
6.1.17	Object 60F2 _h : Positioning Option Code	81
6.2	How to move a Motor in pp Mode	82
7	Profile velocity mode	83
7.1	Detailed Object Specifications	83
7.1.1	Object 6040 _h : Control Word	83
7.1.2	Object 6041 _h : Status Word	84
7.1.3	Object 6062 _h : Position Demand Value	86
7.1.4	Object 6063 _h : Position Actual Internal Value	86
7.1.5	Object 6064 _h : Position Actual Value	87
7.1.6	Object 6065 _h : Following Error Window	87
7.1.7	Object 606C _h : Velocity Actual Value	88
7.1.8	Object 607D _h : Software Position Limit	88
7.1.9	Object 6083 _h : Profile Acceleration	89
7.1.10	Object 6085 _h : Quick Stop Deceleration	89
7.1.11	Object 60FF _h : Target Velocity	89
7.2	How to move a Motor in pv Mode	90



8 Homing mode	91
8.1 Homing Methods	92
8.1.1 Homing Method 1: Homing on negative Limit Switch and Index Pulse	92
8.1.2 Homing Method 2: Homing on positive Limit Switch and Index Pulse	93
8.1.3 Homing Method 3: Homing on positive Home Switch and Index Pulse	93
8.1.4 Homing Method 5: Homing on negative Home Switch and Index Pulse	93
8.1.5 Homing Method 17, 18, 19, and 21: Homing without Index Pulse	94
8.1.6 Homing Method 33 and 34: Homing on next Index Pulse	94
8.1.7 Homing Method 35: Current position as home position	95
8.2 Detailed Object Specifications	96
8.2.1 Object 6040 _n : Control Word	96
8.2.2 Object 6041 _n : Status Word	97
8.2.3 Object 606C _n : Velocity Actual Value	98
8.2.4 Object 607C _n : Home Offset	99
8.2.5 Object 6098 _n : Homing Method	100
8.2.6 Object 6099 _n : Homing Speeds	100
8.2.7 Object 609A _n : Homing Acceleration	100
8.2.8 Object 2100 _n : Home Offset Display	101
8.3 How to start a Homing in hm Mode	101
9 Emergency Messages (EMCY)	103
10 Figures Index	105
11 Tables Index	106
12 Supplemental Directives	109
12.1 Producer Information	109
12.2 Copyright	109
12.3 Trademark Designations and Symbols	109
12.4 Target User	109
12.5 Disclaimer: Life Support Systems	109
12.6 Disclaimer: Intended Use	109
12.7 Collateral Documents & Tools	110
13 Revision History	111
13.1 Firmware Revision	111
13.2 Document Revision	111



1 Preface

This document specifies objects and modes of operation of the Trinamic TMCM-3212 stepper 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-3212 TMCL firmware again through the USB interface, with the help of the firmware update function of the TMCL-IDE 3.0.

1.1 General Features of this CANopen Implementation

Main Characteristics

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

SDO Communication

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

PDO Communication

- Producer
- Consumer
- RPDOs
 - Axis 0: 1, 2, 3, 4
 - Axis 1: 65, 66, 67, 68
 - Axis 2: 129, 130, 131, 132
 - Transmission modes: asynchronous.
 - Dynamic mapping with max. 3 mapping entries.
 - Default mappings: according to CiA-402 for first three PDOs of each axis, manufacturer specific for other PDOs of each axis.
- TPDOs
 - Axis 0: 1, 2, 3, 4
 - Axis 1: 65, 66, 67, 68
 - Axis 2: 129, 130, 131, 132
 - Transmission modes: asynchronous, asynchronous with event timer, synchronous.
 - Dynamic mapping with max. 3 mapping entries.
 - Default mappings: according to CiA-402 for first three PDOs of each axis, manufacturer specific for other PDOs of each axis.



Further Characteristics

- SYNC: consumer (TPDOs 3, 67, 131 are synchronous PDOs)
- Emergency: producer
- RTR: supported only for node guarding/life guarding
- Heartbeat: consumer and producer

1.2 Abbreviations used in this Manual

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

Table 1: Abbreviations used in this Manual

1.3 Firmware Update

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



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

1.4.1 stallGuard2

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

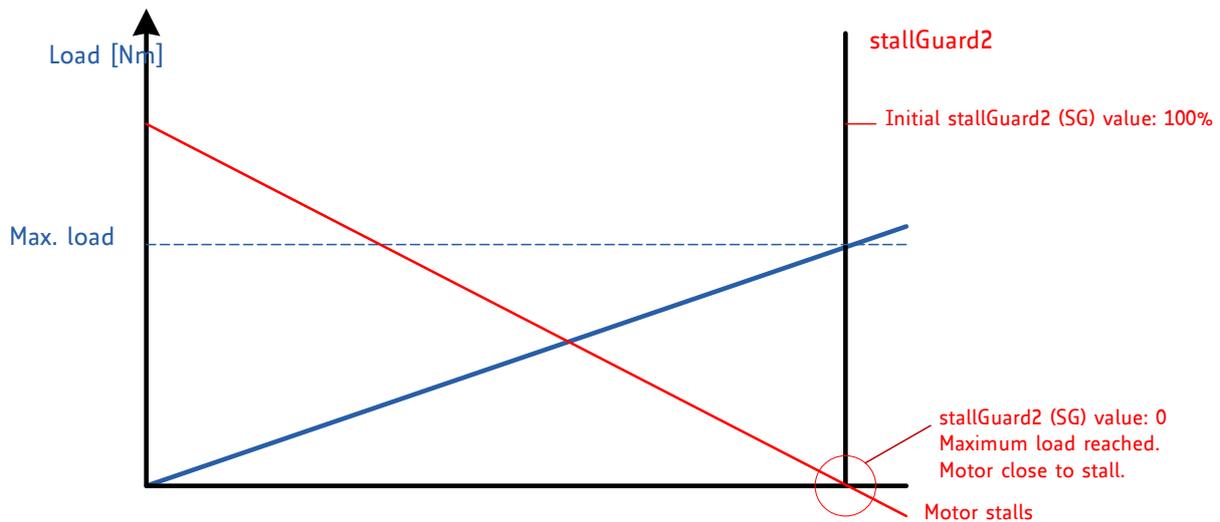


Figure 1: stallGuard2 Load Measurement as a Function of Load

1.4.2 coolStep

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



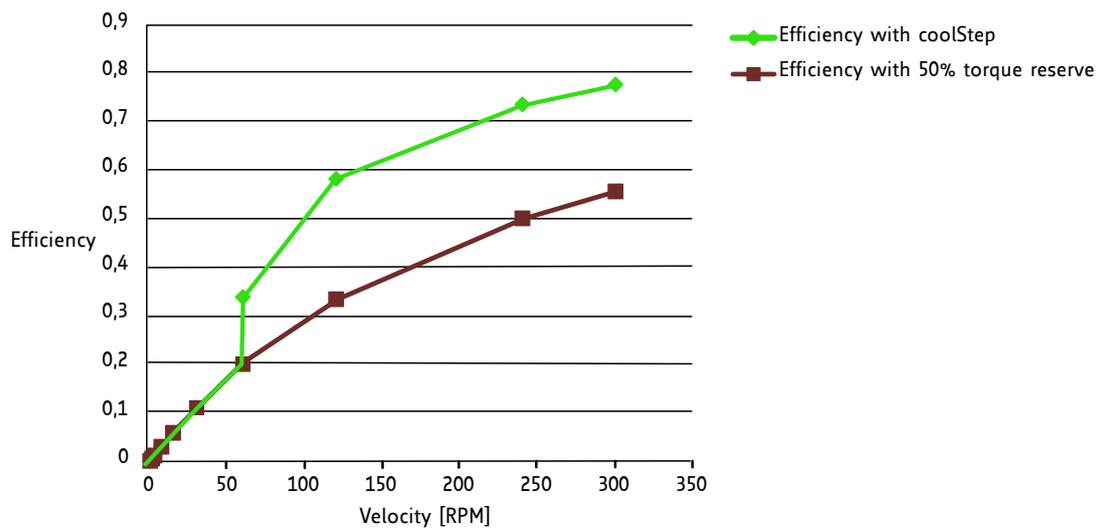


Figure 2: Energy Efficiency Example with coolStep



2 Communication

2.1 Reference Model

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

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

The application and the application layer interact with service primitives.

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

Table 2: Service Primitives

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



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

Table 3: Service Types



2.2 NMT State Machine

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

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

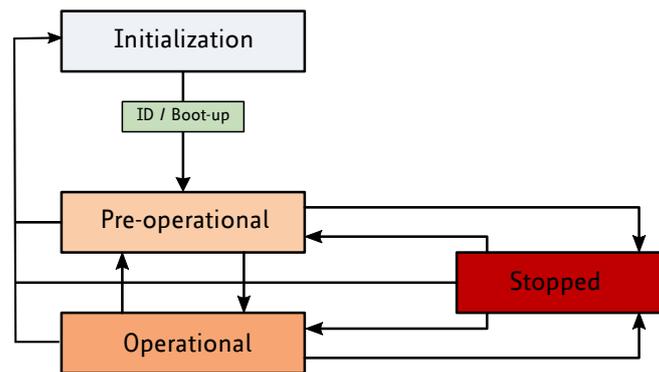


Figure 3: NMT State Machine

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

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

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

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

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

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



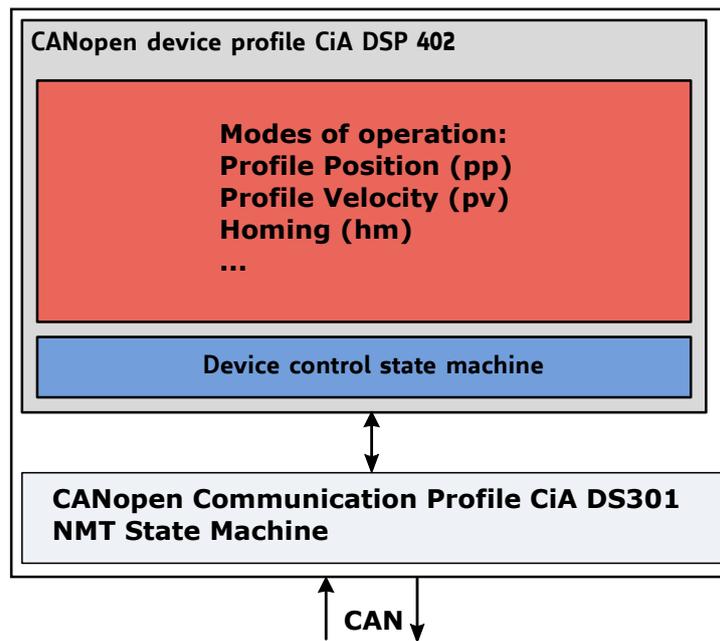


Figure 4: Communication Architecture

2.3 Device Model

A CANopen device mainly consists of the following parts:

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



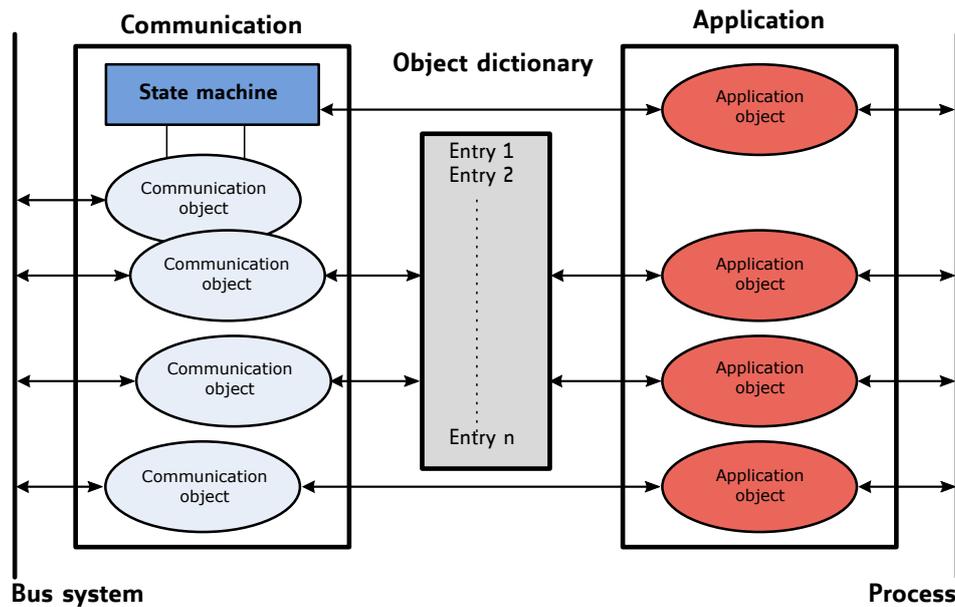


Figure 5: Device Model

2.4 Object Dictionary

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

Object Dictionary	
Index	Object
0000 _h	Not used.
0001 _h – 001F _h	Static data types.
0020 _h – 003F _h	Complex data types.
0040 _h – 005F _h	Manufacturer specific complex data types.
0060 _h – 007F _h	Device profile specific static data types.
0080 _h – 009F _h	Device profile specific complex data types.
00A0 _h – 0FFF _h	Reserved for further use.
1000 _h – 1FFF _h	Communication profile area.
2000 _h – 5FFF _h	Manufacturer specific profile area.
6000 _h – 9FFF _h	Standardized device profile area.
A000 _h – BFFF _h	Standardized interface profile area.
C000 _h – FFFF _h	Reserved for further use.

Table 4: Object Dictionary



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

The manufacturer segment at indices 2000_h through 5FFF_h contains manufacturer specific objects. These objects control the special features of the Trinamic TMCM-3212 motion control device.

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

2.4.1 Object Indices on Multi-Axis Modules

On a multi-axis module like the TMCM-3212 each object in the manufacturer area and each object in the profile specific area is available for each motor. In this manual, only the object indices for motor #0 are shown. The objects for the other motors can be accessed by adding offsets to the object indices:

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

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

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

Table 5: Multi-Axis Object Indices



3 Communication area

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

3.1 Detailed object specifications

3.1.1 Object 1000_h: Device Type

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

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

Table 6: Object Description (1000_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	UNSIGNED32	FFFC0192 _h

Table 7: Entry Description (1000_h)

3.1.2 Object 1001_h: Error Register

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

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

Table 8: Object Description (1001_h)

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

Table 9: Entry Description (1001_h)



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

Table 10: Error Register Bits

3.1.3 Object 1005_h: COB-ID SYNC Message

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

Value Definition		
Bit	Name	Definition
30	Generate	0: Device does not generate SYNC message 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 11: Value Definition (1005_h)

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

Table 12: Object Description (1005_h)

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

Table 13: Entry Description (1005_h)

3.1.4 Object 1008_h: Manufacturer Device Name

This object contains the manufacturer device name.

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

Table 14: Object Description (1008_h)

Entry Description				
Sub-index	Access	PDO Mapping	Value Range	Default Value
0	ro	no	—	TMCM-3212

Table 15: Entry Description (1008_h)

3.1.5 Object 1009_h: Manufacturer Hardware Version

This object contains the hardware version description.

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

Table 16: Object Description (1009_h)

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

Table 17: Entry Description (1009_h)

3.1.6 Object 100A_h: Manufacturer Software Version

This object contains the software version description.

Object Description			
Index	Name	Object Type	Data Type
100A _h	Manufacturer Software Version	Variable	Visible String

Table 18: Object Description (100A_h)



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

Table 19: Entry Description (100A_h)

3.1.7 Object 100C_h: Guard Time

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

Object Description			
Index	Name	Object Type	Data Type
100C _h	Guard Time	Variable	UNSIGNED16

Table 20: Object Description (100C_h)

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

Table 21: Entry Description (100C_h)

3.1.8 Object 100D_h: Life Time Factor

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

Object Description			
Index	Name	Object Type	Data Type
100D _h	Life Time Factor	Variable	UNSIGNED8

Table 22: Object Description (100D_h)

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

Table 23: Entry Description (100D_h)

3.1.9 Object 1010_h: Store Parameters

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



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

There are several parameter groups:

- Sub-index 0_h: contains the largest sub-index that is supported.
- Sub-index 1_h: saves all parameters.
- Sub-index 2_h: saves communication parameters 100C_h, 100D_h, 1015_h, 1017_h, and 1029_h.
- Sub-index 3_h: saves device profile parameters.
- Sub-index 4_h: saves motor 0 parameters.
- Sub-index 5_h: saves motor 1 parameters.
- Sub-index 6_h: saves motor 2 parameters.

Note In order to avoid storage of parameters by mistake, storage is only executed when a specific signature is written to the appropriate sub-Index. This signature is "save" (65766173_h, see also table 24).

Save Signature			
e	v	a	s
65 _h	76 _h	61 _h	73 _h

Table 24: Save Signature

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

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

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

Table 25: Object Description (1010_h)



Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
01h	Save all parameters	rw	no	UNSIGNED32	—
02h	Save communication parameters	rw	no	UNSIGNED32	—
03h	Save device profile parameters	rw	no	UNSIGNED32	—
04h	Save motor axis 0 parameters	rw	no	UNSIGNED32	—
04h	Save motor axis 1 parameters	rw	no	UNSIGNED32	—
04h	Save motor axis 2 parameters	rw	no	UNSIGNED32	—

Table 26: Entry Description (1010_h)

3.1.10 Object 1011_h: Restore Parameters

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

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

There are several parameter groups:

- Sub-index 0_h: contains the largest sub-index that is supported.
- Sub-index 1_h: restores all parameters.
- Sub-index 2_h: restores communication parameters 100C_h, 100D_h, 1015_h, 1017_h, and 1029_h.
- Sub-index 3_h: restores device profile parameters.
- Sub-index 4_h: restores motor 0 parameters.
- Sub-index 5_h: restores motor 1 parameters.
- Sub-index 6_h: restores motor 2 parameters.

Note In order to avoid restoring the parameters by mistake, restoring is only executed when a specific signature is written to the appropriate sub-Index. This signature is "load" (64616F6C_h, see also table 27).

Load Signature			
d	a	o	l
64 _h	61 _h	6F _h	6C _h

Table 27: Load Signature



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

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

After the default values have been restored they will become active after the next rest or power cycle of the TMC3212.

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

Table 28: Object Description (1011_h)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
01h	Restore all parameters	rw	no	UNSIGNED32	—
02h	Restore communication parameters	rw	no	UNSIGNED32	—
03h	Restore device profile parameters	rw	no	UNSIGNED32	—
04h	Restore motor axis 0 parameters	rw	no	UNSIGNED32	—
04h	Restore motor axis 1 parameters	rw	no	UNSIGNED32	—
04h	Restore motor axis 2 parameters	rw	no	UNSIGNED32	—

Table 29: Entry Description (1011_h)

3.1.11 Object 1014_h: COB-ID Emergency Object

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

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

Table 30: Object Description (1014_h)

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

Table 31: Entry Description (1014_h)



3.1.12 Object 1015_h: Inhibit Time EMCY

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

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

Table 32: Object Description (1015_h)

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

Table 33: Entry Description (1015_h)

3.1.13 Object 1016_h: Consumer Heartbeat Time

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

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

Table 34: Value Definition (1016_h)

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

Table 35: Object Description (1016_h)



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

Table 36: Entry Description (1016_h)

3.1.14 Object 1017_h: Producer Heartbeat Time

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

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

Table 37: Object Description (1017_h)

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

Table 38: Entry Description (1017_h)

3.1.15 Object 1018_h: Identity Object

The object 1018_h contains general information about the device:

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

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

Table 39: Object Description (1018_h)

Entry Description					
Sub-index	Description	Access	PDO Mapping	Value Range	Default Value
00 _h	Number of entries	ro	no	0...3	3
01 _h	Vendor ID	ro	no	UNSIGNED32	0286 _h
02 _h	Product code	ro	no	UNSIGNED32	3212
03 _h	Revision number	ro	no	UNSIGNED32	e.g. 20003 _h for version 2.3

Table 40: Entry Description (1018_h)

3.1.16 Object 1029_h: Error Behaviour

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

- Communication error
- Application error

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

Table 41: Object Description (1029_h)

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

Table 42: Entry Description (1029_h)

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

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

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

