

PCAN-MicroMod CANopen

CANopen® Firmware for PCAN-MicroMod

User Manual v1.1.1



CANopen®

PEAK
System

Products taken into account

Product name	Model	Part number
CANopen® Firmware for PCAN-MicroMod		

CANopen® and CiA® are registered community trade marks of CAN in Automation e.V.

All other product names mentioned in this document may be the trademarks or registered trademarks of their respective companies. They are not explicitly marked by "™" and "®".

© 2012 PEAK-System Technik GmbH

PEAK-System Technik GmbH
Otto-Roehm-Strasse 69
64293 Darmstadt
Germany

Phone: +49 (0)6151 8173-20
Fax: +49 (0)6151 8173-29

www.peak-system.com
info@peak-system.com

Document version 1.1.1 (2012-02-24)

Contents

1	Outline and Introduction	5
1.1	About the CANopen® Implementation	6
1.2	Scope of Supply	6
1.3	Hardware and Software Requirements	6
1.4	Terminology and Abbreviations	7
2	Product and Firmware Selection	13
2.1	Device Summary	13
2.2	Loading the Firmware into a MicroMod	14
2.3	Processing Times and Performance	15
3	Hardware Features	18
3.1	Status and Error LEDs	18
3.2	Boot Mode Selection with Solder Bridges	19
3.3	Assignment of Bit Rate and Node ID	19
3.4	Status Info on Serial Port	20
4	CANopen® Implementation	22
4.1	Product Identification	23
4.2	Default Connection Set	24
4.3	Default PDO Mapping	25
4.4	Input Processing	26
4.5	Output Processing	28
4.6	Heartbeat Production and Consumption	29
4.7	Storing Parameters	30

Appendix A Object Dictionary Reference	32
A.1 Overview	32
A.2 Communication Profile Entries	34
A.3 Device Profile Entries	50

1 outline and Introduction

CANopen® is a popular industrial communication standard used in many embedded networking applications such as mobile machinery, industrial automation, medical equipment and many more. The standards including all the device and application profiles are maintained by the CiA, the CAN in Automation User's and Manufacturer's Group.

One of the main reasons for the popularity of CANopen is its flexibility towards customization. CANopen provides a set of standardized communication functions of which only a little percentage is mandatory. When designing a network for a specific application, engineers and designers can choose which functions are needed and can even add custom functionality where required.

Nevertheless, the standardized CANopen device and application profiles available still enable off-the-shelf, plug-and-play usage of CANopen devices. System designers that need to add standardized sensors or actuators such as generic I/Os, encoders, drives, and motors can choose from a wide variety of products from numerous manufacturers.

Although this user manual explains all the CANopen features implemented by the software provided, it still assumes that the reader has some basic knowledge about Controller Area Network (CAN) and CANopen. If in doubt, the CAN and CANopen specifications or books like "Embedded Networking with CAN and CANopen" (www.canopenbook.com) should be consulted.



Note: This user manual only contains information about the CANopen firmware to be used with a PCAN-MicroMod. For detailed information about the hardware please refer to the extra manual.

1.1 About the CANopen® Implementation

The PCAN-MicroMod I/O modules fall into the category of off-the-shelf CANopen generic I/O devices. The CANopen software package for these modules implements the following CiA standards:

- └ CiA® 301 "CANopen Application Layer and Communication Profile" version 4.02
- └ CiA® 401 "CANopen Device Profile for Generic I/O Modules" version 2.1

By loading the appropriate firmware into the modules, they can be directly used as standardized CANopen generic I/O modules.

The EDS files (Electronic Data Sheet) provided along with the firmware were tested with their corresponding module for CANopen conformance using the official CANopen conformance tests.

1.2 Scope of supply

The CANopen firmware package is license free for all users of PCAN-MicroMod and can be downloaded as part of the PCAN-MicroMod package from www.peak-system.com.

The ZIP file available online contains the latest version of the firmware (Fujitsu hex file format), the EDS files, and the user manual in PDF format. Furthermore, a flash tool for loading the firmware into a MicroMod is included.

1.3 Hardware and Software Requirements

The CANopen firmware is loaded into a PCAN-MicroMod using the PCAN-MicroMod Evaluation Board. With the flash programming utility, the hex files provided can be programmed.

In order to configure a MicroMod a generic CANopen Configuration Tool with access to the CAN bus is required. We recommend the software package PCANopen Magic Pro (www.canopenmagic.com) with a PCAN-USB interface.

1.4 Terminology and Abbreviations

The following CANopen terms and abbreviations are used in this manual without further explanation. Please refer to the CANopen standards CiA[®] 301, CiA[®] 302, and CiA[®] 401 or books like “Embedded Networking with CAN and CANopen” for further explanation.

[xxxxh,yyh]

This syntax is used to indicate an Object Dictionary entry using hexadecimal values. The first number ‘xxxx’ represents the Index and ‘yy’ the Subindex.

Bit Rate

The default bit rates supported by many CANopen nodes are 1 Mbit/s, 800 kbit/s, 500 kbit/s, 250 kbit/s, 125 kbit/s, and 50 kbit/s. The bit rate can be changed using the [1F50h,03h] Object Dictionary entry.

Boot-up Message

After a reset, a CANopen node transmits its boot-up message. The message (CAN) identifier used is 700h plus the node’s Node ID number. The boot-up message contains one data byte which is zero.

CANopen Manager

In a CANopen network the optional CANopen Manager typically includes a NMT Master and a Configuration Manager. The NMT Master is responsible for generating the NMT Master Message and starting or stopping the network. The Configuration Manager keeps

track of all nodes connected and ensures that the connected nodes are configured correctly.

Change-of-state (COS)

Generic I/O modules typically monitor their inputs for changes. CANopen nodes can be configured to only transmit this input data, if a change was detected.

Connection Object Identifier, COB-ID

The COB-ID specifies the CAN message details used for a specific connection. These include a (CAN) Message Identifier and typically an enable/disable bit.

Default Connection Set

The default connection set determines which (CAN) Message Identifier is used for which purpose in a CANopen network. Typically (CAN) Message Identifiers are assigned using a base identifier (such as 180h for TPDO1) and adding the Node ID to that base identifier.

Emergency, EMCY

CANopen emergency messages have a default (CAN) Message Identifier of 80h plus the Node ID of the node generating the emergency. Emergency messages contain a CANopen specified emergency code, the error register [1001h,00h] and a manufacturer specific emergency code.

Event Time

For TPDOs the Event Time specifies the periodic time at which TPDOs get transmitted. The Event Time is specified in multiples of milliseconds and typically only used if the Transmission Type is set to 255d/FFh.

Heartbeat Producer and Consumer

In CANopen the recommended function to monitor the health of network nodes is to use the heartbeat mechanism. Each CANopen

node can individually produce a heartbeat message, using the (CAN) Message Identifier 0700h plus the node's Node ID. The message contains a single byte representing the current NMT state of the node (like Pre-Operational, Operational, Stopped). The Heartbeat Producer time [1017h,00h] is specified in multiples of milliseconds. CANopen nodes may have zero, one, or multiple Heartbeat Consumers [1016h,xxh] that can be configured to monitor heartbeats from other nodes. If a node loses a heartbeat monitored, it typically transmits an emergency message and goes back into the Pre-Operational NMT state.

Inhibit Time

For TPDOs that implement a COS detection, the Inhibit Time is used to prevent back-to-back transmission of input data that changes frequently. The Inhibit Time is selected in multiples of 100s of microseconds and specifies how long a CANopen node must wait before it may transmit a TPDO again. Note that many CANopen implementations do not offer such a high resolution and the Inhibit Time might get rounded up to milliseconds.

Index

An Object Dictionary entry is selected using a 16-bit Index and an 8-bit Subindex.

Layer Setting Services (LSS)

A CANopen standard defining how layer services like bit rate and Node ID can be changed during operation.

Manager/Master

In CANopen there are several Master and Manager services, typically combined into the 'CANopen Manager'.

Message Identifier

Any communication technology used for CANopen (default is CAN – Controller Area Network) must be able to assign a Message Identifier to any message transferred. CAN uses the default of an 11-bit

Message Identifier. The Default Connection Set selects how these Message Identifiers are used.

Network Management (NMT)

Each CANopen node internally implements an NMT Slave state machine with the major states being Pre-Operational (used to configure devices), Operational (actively producing and consuming PDOs) and Stopped (limiting communication to Heartbeats). The NMT Master Message is used to command nodes to switch their NMT States.

NMT Master Message

The NMT Master (often integrated into the CANopen Manager) uses the NMT Master Message to switch the NMT State of individual or all nodes. The (CAN) Message Identifier of the NMT Master Message is 0 (zero) with 2 data bytes.

Node ID

In any CANopen network each CANopen node must have a unique Node ID in the range of 1 to 127. The Node ID can be changed using the [1F50h,03h] Object Dictionary entry.

Object Dictionary, OD

Each CANopen node internally structures its data (configuration and process data) into an Object Dictionary, similar to a look-up table. Each entry can be identified by a 16-bit Index and an 8-bit Subindex. The CANopen standards specify which entry is used for which purpose. A configuration tool or CANopen Manager can access the OD of any node using SDO communication.

PDO Communication Parameters

The PDO communication parameters include COB ID, Transmission Type, Inhibit Time, and Event Time. They specify the used (CAN) message identifier and for Transmit PDOs the selected trigger events.

PDO Mapping Parameters

The PDO Mapping Parameters select the contents of a PDO. One or multiple variables from the Object Dictionary can be combined into a PDO.

Process Data Object, PDO

These (CAN) messages are used for communicating process data.

Receive Process Data Object, RPDO

Each CANopen node typically can transmit and receive PDOs. To better distinguish between the services, the terms RPDOs and TPDOs are used. Note that a TPDO of one node is a RPDO for all other nodes that consume this PDO.

Service Data Object, SDO

These (CAN) messages are used for communicating service/configuration data.

Subindex

An Object Dictionary entry is selected using a 16-bit Index and an 8-bit Subindex.

Synchronized Communication

In synchronized communication mode (one of the Transmission Types supported by CANopen), RPDO data received is not immediately applied to the outputs; it is only applied after the SYNC message was received. Synchronized TPDO data gets transmitted right after the SYNC message is received.

Transmission Type

For each PDO the Transmission Type can be selected as it is one of the PDO Communication Parameters. The default Transmission Type is 255/FFh selecting the communication specific for the Device Profile. In the case of CiA® 401 Generic I/O this means that both Event Time and Inhibit Time are used.

Transmit Process Data Object, TPDO

Each CANopen node typically can transmit and receive PDOs. To better distinguish between the services, the terms RPDOs and TPDOs are used. Note that a TPDO of one node is a RPDO for all other nodes that consume this PDO.

2 Product and Firmware Selection

All CANopen firmware versions for PCAN-MicroMod use the same code basis and thus have similar features. The main difference is that the number of I/O channels made available matches the hardware layout of the various motherboards.

2.1 Device Summary


The generic firmware version (device 0) is a superset, as it implements the maximum number of I/O channels supported. This version is intended for users that embed a MicroMod into their own customized hardware.

All other firmware versions provided implement a specific subset enabling a limited number of I/O channels.

The following table lists the number of channels available with each device. The header indicates the resolution for each I/O channel (exceptions apply for Analog 2).


Device		DI (1 bit)	DO (1 bit)	AI (10 bits)	AO (8 bits)
0	MicroMod (custom) Evaluation Board Mix 3	8	7 (see also note below)	8	4
1	Digital 1	8	4	0	1
2	Digital 2	8	4	0	1
3	Analog 1	8 (shared AI)	0	8	4
4	Mix 1	6	0	4	2
5	Mix 2	2	1	5	2
6	Analog 2	0	0	8 (16 bits)	4 (12 bits)

Resolution (bits)	Value range hex	Value range dec
8	0 - 00FFh	0 - 255
10	0 - 03FFh	0 - 1023
12	0 - 0FFFh	0 - 4095
16 signed	0 - FFFFh	-32768 - +32767

 **Note for Device 0:** The digital outputs of the PCAN-MicroMod Evaluation Board and the motherboard Mix 3 behave differently regarding the Low and High states. Please refer to the user manual of the respective motherboard.

2.2 Loading the Firmware into a MicroMod

The hex files provided can be loaded into a MicroMod using the PCAN-MicroMod Evaluation Board and the Fujitsu Flash 16 programming utility (part of the PCAN-MicroMod software package).

 Follow these steps to upload the firmware:

1. Make sure that the power plug of the Evaluation Board is disconnected.
2. Plug the MicroMod to be programmed onto the Evaluation Board.
3. On the Evaluation Board set the programming jumper S5 to "Prog" (2-3) to activate the on-chip boot loader.
4. Make sure that the DIP switches 1 and 2 are in the 'off' position.
5. Connect the serial connector "RS232" to the COM port of the PC.
6. Start the Fujitsu Flash 16 programming utility (`flash.exe`).

7. Re-apply power to the Evaluation Board and press the reset button to ensure the processor goes properly into the boot mode.
8. Make the following settings in the Fujitsu Flash 16 utility:
 - Target Microcontroller: MB90F497/G
 - Crystal Frequency: 4 MHz
 - Hex File: one of the hex files implementing CANopen
9. Press the button “Full Operation”.

The Flash 16 utility converts the hex file, transfers it to MicroMod and programs it into the flash memory.
10. Once the process is completed, you can power down the Evaluation Board, remove the MicroMod, and insert it into appropriate CANopen target hardware.

2.3 Processing Times and Performance

The CANopen firmware was optimized to provide the best possible performance. Nevertheless, it should be noted that in any CANopen based I/O system the flexible configuration potentially allows configuring a module in such a way that its performance can vary considerably.

As a result all timings given here are not ‘true’ worst-case timings. For example, they can be delayed by higher-priority CAN activity, such as an NMT Master message.

On an 1-Mbit/s CAN network, Event, Inhibit or synchronization cycle times of down to 5 milliseconds may be used with the MicroMod without any performance penalty. Even shorter times are possible but their accuracy depends on the overall PDO configuration (how many PDOs need to be that fast and how long are they).

2.3.1 Digital I/O Performance

Processing of all digital I/Os happens on a 500-microsecond timer basis. For best performance, digital I/Os should always be mapped into the first RPDO/TPDO as that is processed with a higher priority.

Digital outputs get applied to the outputs within one millisecond after the triggering message was received. This is either the RPDO1 itself or the SYNC message if synchronized communication is used.

With the change-of-state (COS) detection a change in the digital inputs is typically recognized within one millisecond and immediately triggers the transmission of TPDO if COS communication is used. However, process data received is processed at a higher priority, potentially delaying the COS detection.

2.3.2 Analog I/O Performance

Processing of all analog I/Os happens on a 1-millisecond timer basis. Each millisecond one analog input channel and one analog output channel are updated. As a result, the overall delay in analog data processing depends on the number of analog channels used.

With a system using 8 analog inputs each input is updated every 8 milliseconds. The COS and minimal delta detection typically requires another millisecond to execute and detect potential changes.

If a system has 4 analog outputs, then the delay in applying new received values to the outputs can be up to 4 milliseconds.

2.3.3 Background Task and Change Of State Detection

Processing of change-of-state and analog input delta detection, heartbeat production and consumption and SDO request processing is all handled in the main background task. As both CAN Receive Interrupt and the 500-microsecond Timer Interrupt have a higher priority, the background task can get interrupted and delayed at any time.

The PEAK Status Register [1002h,00h] in the Object Dictionary contains the current average execution time of a single background loop. In operational mode, this number typically varies from 300 to 600 microseconds.

3 Hardware Features

This chapter summarizes the most important PCAN-MicroMod hardware features as far as the CANopen implementation is concerned. For further hardware details like pin assignment and I/O characteristics, please refer to the PCAN-MicroMod user manual and those for the PCAN-MicroMod motherboards.

3.1 Status and Error LEDs

CANopen supports two LED indicators as specified by document DR303 "Indicator Specification". Concerning the CANopen Firmware the RUN LED is located directly on the MicroMod. The ERR LED is related to DO7.

3.1.1 RUN LED

The RUN LED shows one of the following patterns:

LED Status	Description
On	Node is in NMT state "Operational"
Blinking	Node is in NMT state "Pre-operational"
Single Flash	Node is in NMT state "Stopped"

3.1.2 ERR LED

The ERR LED shows one of the following patterns:

LED Status	Description
Off	No Error
On	Node is not yet initialized after reset or a fatal error occurred
Double Flash	Heartbeat lost

3.2 Boot Mode Selection with Solder Bridges

PCAN-MicroMod features a total of 5 solder bridges. The CANopen implementations currently only use the solder bridge '4' as a boot mode indicator. If solder bridge '4' is closed, after reset, PCAN-MicroMod starts in a default boot mode using a CAN bit rate of 125 kbit/s and the CANopen node ID 40h/64d.

A "blinking" LED blinks with a period of 200 milliseconds.

A "single flash" LED is on for 200 milliseconds and then off for 1 second.

3.3 Assignment of Bit Rate and Node ID

The CAN bit rate and CANopen node ID used cannot be changed by hardware. They can be changed by writing specific values to an Object Dictionary entry.

Using a CANopen configuration tool (such as CANopen Magic), writes to the Object Dictionary entry [1F50h,03h] can change the bit rate and node ID.

Writing the 4-byte ASCII string "BPSx" changes the bit rate. 'x' must be replaced with one of the following digits (LSS compatible values):

Value	Bit rate
0	1 Mbit/s
1	800 kbit/s
2	500 kbit/s
3	250 kbit/s
4	125 kbit/s
6	50 kbit/s

Other bit rates are not supported.

Writing the 4-byte ASCII string "Nlxy" changes the node ID. 'xy' must be replaced by a 2-digit hexadecimal number specifying the new node ID, which must be in the range of 01h to 7Fh.



Note: The new settings will only be activated after the next reset of the MicroMod.

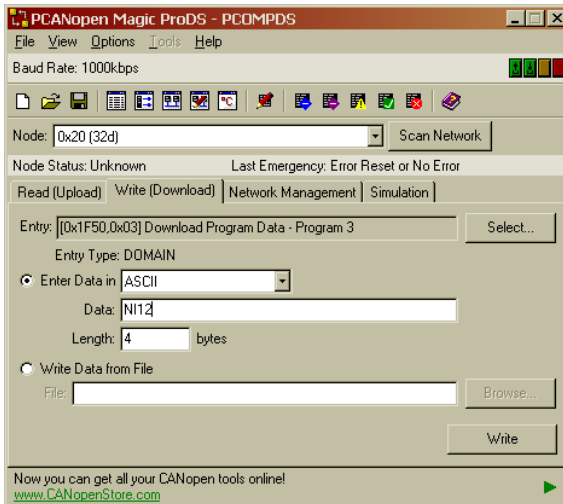


Figure 1: Example for sending the 'NI' command to a node using PCANopen Magic Pro

3.4 Status Info on Serial Port

If a MicroMod is used in hardware with serial line drivers (such as the PCAN-MicroMod Evaluation Board), then some basic status info can be displayed on a terminal. The communication settings are: 19200 bit/s, 8 bits, 1 stop bit, no parity.

Information is only sent during initialization and in case of errors. Information displayed on the terminal includes the CAN bit rate, the CANopen node ID and the number of input and output channels used.

In case of a fatal error, an internal error number is displayed.

The following is a sample output:

```
PEAK MicroMod CANopenIA V1.01 Rev.204
J0-J4: 0
Regular Operation as MicroMod: 125kbps, node ID 32
Number of I/O: DI 1, DO 1, AI 8, AO 4
Initialization completed!
```

4 CANopen® Implementation

The implementation features the major CANopen services as standardized in the following CiA documents:

- CiA® 301 “CANopen Application Layer and Communication Profile” version 4.02
- CiA® 401 “CANopen Device Profile for Generic I/O Modules” version 2.1

In summary, these services are:

- One SDO server giving CANopen Managers and configuration tools access to the internal Object Dictionary (OD)
- Heartbeat Producer functionality allowing the devices to produce their individual heartbeat messages at a configurable rate
- Heartbeat Consumer functionality with up to three channels allowing the monitoring of up to 3 heartbeats
- Although not recommended to be used, the Node Guarding functionality is still implemented in order to be compatible with legacy devices
- Emergency Producer to inform other nodes over major error or other emergency situations
- Re-configuration of Node ID and bit rate during operation
- Storage of configuration parameters in non-volatile memory
- 4 Receive Process Data Objects (RPDO) with dynamic communication and mapping parameters – RPDO configuration including CAN message ID used and contents mapped to the RPDO can be changed during operation

- └ 4 Transmit Process Data Objects (TPDO) with dynamic communication and mapping parameters – TPDO configuration including CAN message ID used, message trigger mechanisms and contents mapped to the TPDO can be changed during operation
- └ Up to 8 digital input channels with configurable polarity inversion
- └ Up to 7 digital output channels with configurable polarity inversion and default output values for error situations
- └ Up to 8 analog input channels with configurable minimal difference (delta) detection
- └ Up to 4 analog output channels with individual default output values for error situation

4.1 Product Identification

There are several Object Dictionary (OD) entries implemented that can be used to identify the PCAN-MicroMod and the specific CANopen firmware used:

- └ [1000h,00h] Device Type
32-bit device type info, see table below
- └ [1008h,00h] Manufacturer Device Name
4-byte ASCII string, see table below
- └ [1009h,00h] Manufacturer Hardware Version
4-byte ASCII string with the hardware version number
- └ [100Ah,00h] Manufacturer Software Version
4-byte ASCII string with the firmware revision number
- └ [1018h,01h] CANopen vendor ID
32-bit vendor ID of PEAK: 00000175h

- └ [1018h,02h] Product Code
32-bit product code, see table below
- └ [1018h,03h] Revision Number
32-bit revision number

Device		Device Type OD [1000h,00h]	Device Name [1008h,00h]	Product ID [1018h,0]
0	MicroMod (custom) Evaluation Board Mix 3	000F0191h	"PCO0"	00100000h
1	Digital 1	000B0191h	"PCO1"	00100001h
2	Digital 2	000B0191h	"PCO2"	00100002h
3	Analog 1	000D0191h	"PCO3"	00100003h
4	Mix 1	000D0191h	"PCO4"	00100004h
5	Mix 2	000F0191h	"PCO5"	00100005h
6	Analog 2	000C0191h	"PCO6"	00100006h

4.2 Default Connection Set

The CANopen implementations use the Default Connection Set. This means the following CAN Message Identifiers are used, unless the devices are re-configured.

CAN Message ID	Direction	Default Usage
000h	Receive	NMT Master Message
080h	Receive	SYNC Message
080h + Node ID	Transmit	Emergency Message
180h + Node ID	Transmit	TPDO 1
200h + Node ID	Receive	RPDO 1
280h + Node ID	Transmit	TPDO 2
300h + Node ID	Receive	RPDO 2
380h + Node ID	Transmit	TPDO 3
400h + Node ID	Receive	RPDO 3
480h + Node ID	Transmit	TPDO 4

CAN Message ID	Direction	Default Usage
500h + Node ID	Receive	RPDO 4
580h + Node ID	Transmit	SDO Response
600h + Node ID	Receive	SDO Request
700h + Node ID	Transmit	Boot-up, Heartbeat

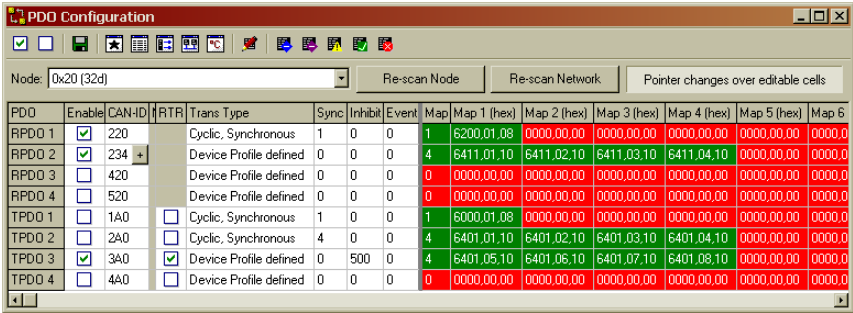
Per default unused PDOs are disabled.

4.3 Default PDO Mapping

The default PDO mapping used is that of the CANopen Device Profile CIA® 401 for generic I/O. A maximum of four variables are mapped to each PDO. The following table illustrates the mapping used by the PCAN-MicroMod device zero (stand-alone module). The other device types only have those variables mapped that are implemented on these devices.

PDO	No. of Entries	Entry 1	Entry 2	Entry 3	Entry 4
RPDO1	1	DO (1-7)	<i>empty</i>	<i>empty</i>	<i>empty</i>
RPDO2	4	AO 1	AO 2	AO 3	AO 4
TPDO1	1	DI (1-8)	<i>empty</i>	<i>empty</i>	<i>empty</i>
TPDO2	4	AI 1	AI 2	AI 3	AI 4
TPDO3	4	AI 5	AI 6	AI 7	AI 8

The PDO parameters can easily be changed with CANopen configuration tools such as PCANopen Magic Pro.



PDO	Enable	CAN-ID	RTR	Trans Type	Sync	Inhibit	Event	Map	Map 1 (hex)	Map 2 (hex)	Map 3 (hex)	Map 4 (hex)	Map 5 (hex)	Map 6
RPDO 1	<input checked="" type="checkbox"/>	220		Cyclic, Synchronous	1	0	0	1	6200,01,08	0000,00,00	0000,00,00	0000,00,00	0000,00,00	0000,00,00
RPDO 2	<input checked="" type="checkbox"/>	234		Device Profile defined	0	0	0	4	6411,01,10	6411,02,10	6411,03,10	6411,04,10	0000,00,00	0000,00,00
RPDO 3	<input type="checkbox"/>	420		Device Profile defined	0	0	0	0	0000,00,00	0000,00,00	0000,00,00	0000,00,00	0000,00,00	0000,00,00
RPDO 4	<input type="checkbox"/>	520		Device Profile defined	0	0	0	0	0000,00,00	0000,00,00	0000,00,00	0000,00,00	0000,00,00	0000,00,00
TPDO 1	<input type="checkbox"/>	1A0	<input type="checkbox"/>	Cyclic, Synchronous	1	0	0	1	6000,01,08	0000,00,00	0000,00,00	0000,00,00	0000,00,00	0000,00,00
TPDO 2	<input type="checkbox"/>	2A0	<input type="checkbox"/>	Cyclic, Synchronous	4	0	0	4	6401,01,10	6401,02,10	6401,03,10	6401,04,10	0000,00,00	0000,00,00
TPDO 3	<input checked="" type="checkbox"/>	3A0	<input checked="" type="checkbox"/>	Device Profile defined	0	500	0	4	6401,05,10	6401,06,10	6401,07,10	6401,08,10	0000,00,00	0000,00,00
TPDO 4	<input type="checkbox"/>	4A0	<input type="checkbox"/>	Device Profile defined	0	0	0	0	0000,00,00	0000,00,00	0000,00,00	0000,00,00	0000,00,00	0000,00,00

Figure 2: PCANopen Magic Pro displaying the PDO parameters with the possibility to edit each entry directly

4.4 Input Processing

Digital input data is XORed with the polarity settings [6002h] before getting stored in [6000h]. The analog input data goes directly into the array at [6401h].

Using SDO requests sent to the device, all input data can be read at any time by a configuration tool or CANopen manager.

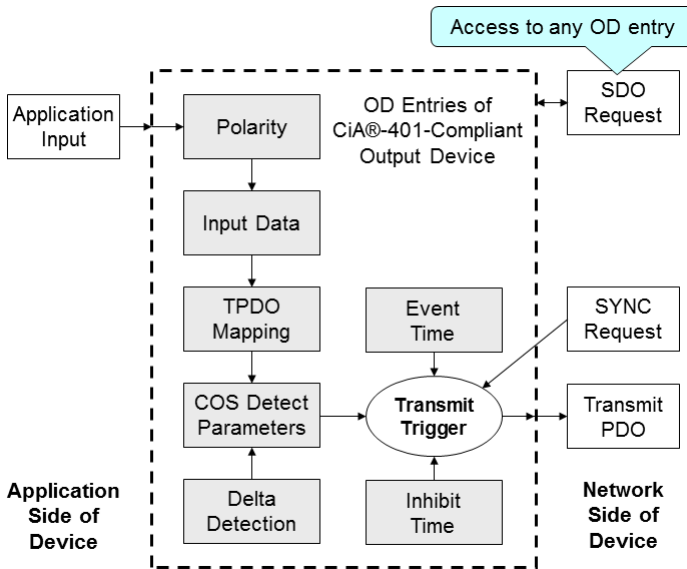


Figure 3: Input data processed by the CANopen firmware

Depending on the TPDO Mapping Parameters [1A0xh], the input data is included in one or multiple TPDOs. Depending on the TPDO Communication Parameters [180xh] the input data gets transmitted if a transmit trigger is activated. For analog inputs, the change-of-state detection includes a delta detection: if the Analog Input Interrupt Enable [6423h] is set, then the array Analog Input Value Difference [6426h] specifies for each channel the minimum difference that a new value must have in comparison to the last transmitted value to trigger a COS event.

4.5 Output Processing

Data received in a RPDO is copied to its destination in the Object Dictionary, depending on the RPDO Mapping Parameters [160xh]. The destination is the array at [6200h] for digital outputs and the array at [6411h] for analog outputs. These entries can also be written to by any CANopen configuration tool or CANopen manager using SDO requests.

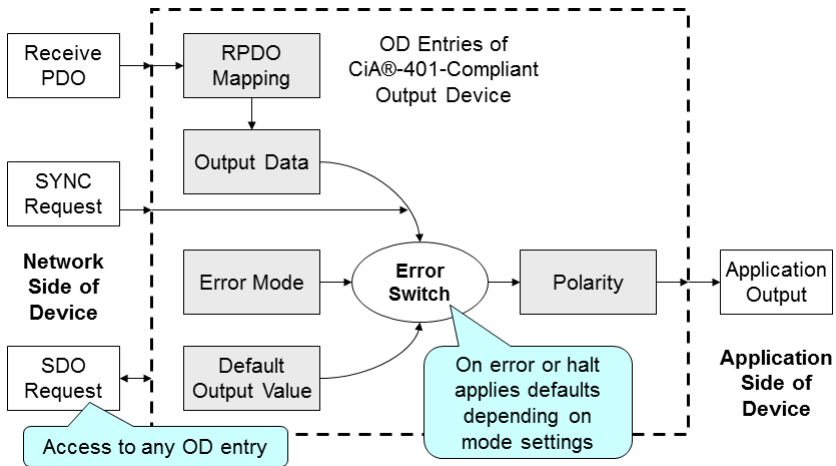


Figure 4: Output data processed by the CANopen firmware

Depending on the RPDO Communication Parameters [140xh] the output data received by RPDO is either applied immediately or upon reception of the next SYNC signal if synchronized communication is used.

Digital output data is XORed with the Polarity Digital Output value at [6202h] before being applied to the output.

When an output channel is marked '1' the Digital Output Error Mode [6206h] array or the Analog Output Error Mode [6443h] array, then default values get applied to the outputs whenever the node leaves

the operational state. The default outputs for each channel are stored in the arrays Digital Output Error Value [6207h] and Analog Output Error Value [6444h].

4.6 Heartbeat Production and Consumption

The recommended method to monitor the health of each CANopen node is using the heartbeat protocol. Each CANopen node produces its own heartbeat. The heartbeat producer time [1017h,00h] can be set with a configuration tool. The time is specified in milliseconds.

In addition, the firmware implements 3 heartbeat consumer channels allowing the MicroMod to monitor up to 3 heartbeats from other nodes. Each heartbeat consumer entry at [1016h,01h-03h] is a Unsigned32 value, where the consumer time (timeout) is stored in bits 0-15, and bits 16-22 hold the node ID of the node monitored.

The heartbeat consumer time must be bigger than the producer time of the node monitored. Otherwise a "heartbeat lost event" will occur almost immediately after monitoring begins. A recommended approach is to make the consumer time 1.5 times bigger than the producer time.

Upon losing a heartbeat, a PCAN-MicroMod device transmits an emergency message and switches itself into the pre-operational mode. When the default error output mode is activated, the default outputs get applied.

4.7 Storing Parameters

All configurable parameters of the CANopen implementation can be stored in non-volatile memory and be automatically activated after the next reset. This way complex configuration does not need to be made over and over again after every reset of the device.

To save the current configuration into non-volatile memory, the 4-byte string “save” needs to be written to OD entry Store Parameters [1010h,01h].

It should be noted that saving the current configuration into non-volatile memory takes longer than other SDO accesses. On some CANopen configuration tools it might be necessary to lengthen the timeout for SDO requests.

The manufacturer default configuration can be restored by writing the 4-byte string “load” to the OD entry Restore Parameters [1011h,01h]. The default settings become active after the next reset.

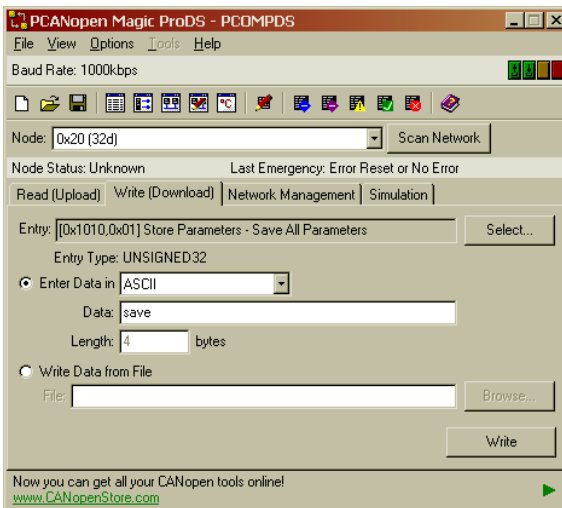


Figure 5: Settings for storing parameters

The OD entries at Verify Configuration [1020h] can be used to save the configuration date and time into non-volatile memory. This allows CANopen managers to verify when the device was configured for the last time.

Appendix A Object Dictionary Reference

A.1 Overview

The firmware implements a CANopen Object Dictionary (OD). The specific OD implementation for each module is specified by the corresponding Electronic Data Sheet (EDS). OD entries belonging to I/O channels are only implemented if the matching device has these channels available.

The following table gives an overview of all OD entries implemented.

Index	Name	Type	Access
1000h	Device Type	Unsigned32	Read Only
1001h	Error Register	Unsigned8	Read Only
1002h	PEAK Status Register	Unsigned32	Read Only
1005h	SYNC COB ID	Unsigned32	Read/Write
1008h	Manufacturer Device Name	Visible_String	Read Only
1009h	Manufacturer Hardware Version	Visible_String	Read Only
100Ah	Manufacturer Software Version	Visible_String	Read Only
100Ch	Guard Time	Unsigned16	Read/Write
100Dh	Life Time Factor	Unsigned8	Read/Write
1010h	Store Parameters	Unsigned32	Read/Write
1011h	Restore Default Parameters	Unsigned32	Read/Write
1016h	Consumer Heartbeat Time	Array of Unsigned32	Read/Write
1017h	Producer Heartbeat Time	Unsigned16	Read/Write
1018h	Identity	Identity Record (0023h)	Read Only
1020h	Verify Configuration	Unsigned32	Read/Write
1400h	1st Receive PDO Parameter	PDO Communication Parameter Record	Read/Write

Index	Name	Type	Access
1401h	2nd Receive PDO Parameter	PDO Communication Parameter Record	Read/Write
1402h	3rd Receive PDO Parameter	PDO Communication Parameter Record	Read/Write
1403h	4th Receive PDO Parameter	PDO Communication Parameter Record	Read/Write
1600h	1st Receive PDO Mapping	PDO Mapping Parameter Record	Read/Write
1601h	2nd Receive PDO Mapping	PDO Mapping Parameter Record	Read/Write
1602h	3rd Receive PDO Mapping	PDO Mapping Parameter Record	Read/Write
1603h	4th Receive PDO Mapping	PDO Mapping Parameter Record	Read/Write
1800h	1st Transmit PDO Parameter	PDO Communication Parameter Record	Read/Write
1801h	2nd Transmit PDO Parameter	PDO Communication Parameter Record	Read/Write
1802h	3rd Transmit PDO Parameter	PDO Communication Parameter Record	Read/Write
1803h	4th Transmit PDO Parameter	PDO Communication Parameter Record	Read/Write
1A00h	1st Transmit PDO Mapping	PDO Mapping Parameter Record	Read/Write
1A01h	2nd Transmit PDO Mapping	PDO Mapping Parameter Record	Read/Write
1A02h	3rd Transmit PDO Mapping	PDO Mapping Parameter Record	Read/Write
1A03h	4th Transmit PDO Mapping	PDO Mapping Parameter Record	Read/Write
1F50h	Download Program	Array of Domain	Read/Write
6000h	Read Digital Inputs	Array of Unsigned8	Read Only
6002h	Polarity Digital Input	Array of Unsigned8	Read/Write
6200h	Write Digital Outputs	Array of Unsigned8	Read/Write
6202h	Polarity Digital Output	Array of Unsigned8	Read/Write
6206h	Digital Output Error Mode	Array of Unsigned8	Read/Write
6207h	Digital Output Error Value	Array of Unsigned8	Read/Write

Index	Name	Type	Access
6401h	Read Analog Inputs	Array of Integer16	Read Only
6411h	Write Analog Outputs	Array of Integer16	Read/Write
6423h	Analog Input Interrupt Enable	Boolean	Read/Write
6426h	Analog Input Value Difference	Array of Unsigned32	Read/Write
6443h	Analog Output Error Mode	Array of Unsigned8	Read/Write
6444h	Analog Output Error Value	Array of Integer32	Read/Write

A.2 Communication Profile Entries

A.2.1 Device Type (1000h)

[1000h,00h], Unsigned32, read only

This entry indicates the number of the device profile used and provides some additional basic information about which features of the device profile are used in the node. The entry value is constructed as follows:

Bit	Description
0-15	Device Profile Number (401d/191h)
16	Set, if the device has digital inputs
17	Set, if the device has digital outputs
18	Set, if the device has analog inputs
19	Set, if the device has analog outputs
20-31	Unused, return zero

A.2.2 Error Register (1001h)

[1001h,00h], Unsigned8, read only, may be mapped to TPDO

The error register value indicates if an error has occurred within the device. The Error Register is included in byte two of the Emergency

object, but may also be mapped into PDOs. The bits in the error register are defined as follows:

Bit	Description
0	Generic Error
1	Current (always zero)
2	Voltage (always zero)
3	Temperature (always zero)
4	Communication Error
5	Device Profile Defined Error (always zero)
6	Reserved (always zero)
7	Manufacturer Specific Error (always zero)

A.2.3 PEAK Status Register (1002h), MCU Load, and Performance

[1002h,00h], Unsigned32, read only

Bits 0 through 11 contain an MCU load indicator. Internally the devices measure the runtime of the main background loop working on all input and output data. The value returned in these bits is the average (measuring 10 loops at the time) runtime of a single loop in microseconds. In operational status this value typically ranges from 400 microseconds to 650 microseconds.

In each loop all digital inputs and outputs are updated and one each of analog inputs and analog outputs.

If a device has 4 analog outputs, then it takes 4 loops to update all analog output channels. So if new analog output data was received via CANopen it can take up to four loop executions until all analog output channels are set to the new value.

A.2.4 SYNC COB ID (1005h)

[1005h,00h], Unsigned32, read/write

This entry contains the CAN ID used by the SYNC message along with a flag to indicate if the node generates the SYNC message or not. In synchronous PDO communication mode the SYNC message triggers the TPDOs transmitted.

The 32 bits of the entry are used as follows:

Bit	Description
0-10	COB ID for SYNC Object
11-28	Set to 0
29	Set to 0 as only 11-bit CAN IDs are supported
30	Set to 0 as the node does not generate the SYNC Object
31	Set to 0

A.2.5 Manufacturer Device Name (1008h)

[1008h,00h], visible_string (4 ASCII bytes), read only

Upon reading, this entry returns a 4-byte ASCII string with the device name. See the table in section 4.1 on page 23 for a list of names used.

A.2.6 Manufacturer Hardware Version (1009h)

[1009h,00h], visible_string (4 ASCII bytes), read only

Upon reading, this entry returns a 4-byte ASCII string with the hardware version number.

A.2.7 Manufacturer Software Version (100Ah)

[1009h,00h], visible_string (4 ASCII bytes), read only

Upon reading, this entry returns a 4-byte ASCII string with the software revision number of the CANopen firmware.

A.2.8 Guard Time (100Ch)

[100Ch,00h], Unsigned16, read/write



Note: The CiA recommends not to use node guarding. This feature has only been implemented to be backwards compatible to legacy devices.

The guard time specifies how long the period should be in ms between node guarding requests sent to the device. If the NMT master implements node guarding, then it should read this entry and transmit the node guarding requests to the node at the frequency indicated by the value of this entry.

If a response from the node to a node guarding request is not transmitted within the node life time, then a node guarding event occurs, indicating that the node may have possibly stopped working. If a node guarding request from the NMT Master is not received within the node life time, then the node knows that the NMT Master may have possibly stopped working.

The node life time is the guard time multiplied by the life time factor.

A.2.9 Life Time Factor (100Dh)

[100Dh,00h], Unsigned8, read/write



Note: The CiA recommends not to use node guarding. This feature has only been implemented to be backwards compatible to legacy devices.

The life time factor specifies the number of multiples of the guard time to wait for a response from the node to a node guarding request.

If a response from the node to a node guarding request is not transmitted within the node life time, then a node guarding event occurs, indicating that the node may have possibly stopped working. If a node guarding request from the NMT Master is not received within the node life time, then the node knows that the NMT Master may have possibly stopped working.

The node life time is the guard time multiplied by the life time factor.

A.2.10 Store Parameters (1010h)

[1010h,00h], unsigned8, read only

Returns 1

[1010h,01h] unsigned32, read/write

The PCAN-MicroMod devices support Subindex one, store all parameters to save all configuration data into non-volatile memory.

By writing to the subentries, the node can be instructed to immediately store all or some of the settings in non-volatile memory.

By reading the subentries, non-volatile storage capabilities of the node may be determined.

By writing the value 65766173h (ASCII "save") to [1010h,01h], the configurable parts of the Object Dictionary are stored in non-volatile memory.

If a value other than "save" is written, or if the parameter storing fails for some reason, an SDO abort message is transmitted.

Reading [1010h,01h] returns '1' indicating that the device can save the parameters by writing to this subentry.

A.2.11 Restore Default Parameters (1011h)

[1011h,00h], Unsigned8, read only

Returns 1

[1011h,01h] Unsigned32, read/write

The [1011h,00h] entry provides a means to restore the manufacturer default values for the configurable parameters of the Object Dictionary. By writing the value 64616F6Ch (ASCII "load") to [1011h,00h], the configurable parameters will be restored to their manufacturer default values on the next reset of the node or the next power cycle.

A.2.12 Consumer Heartbeat Time (1016h)

[1016h,00h], Unsigned8, read only

Returns 3

[1016h,01h], Heartbeat Consumer Channel 1, Unsigned32, read/write

[1016h,02h], Heartbeat Consumer Channel 2, Unsigned32, read/write

[1016h,03h], Heartbeat Consumer Channel 3, Unsigned32, read/write

The devices implement three heartbeat consumer channels and may listen to the heartbeat messages generated by other nodes on the network. Each subentry specifies the maximum time to wait for a heartbeat from a specific node before generating an internal "Heartbeat Lost" event. The Heartbeat Consumer Time is specified in milliseconds. Measurement begins after reception of the first heartbeat message. It does not begin after reception of a boot-up message.

Each Subindex specifies the Heartbeat Consumer Time for one CANopen node. The value of the entry is constructed as follows:

Bit	Description
0-15	Heartbeat consumer time in milliseconds
16-23	Node ID
24-31	Reserved, set to 0

The Heartbeat Consumer Time of a specific node must be higher than the Heartbeat Producer Time of the node. It is recommended to set the consumer time at least 1.5 times higher than the producer time. The Producer Time can be read from entry 1017h.

Specifying a Heartbeat Consumer Time of zero for a specific Node ID disables the heartbeat monitoring of that node.

A.2.13 Producer Heartbeat Time (1017h)

[1017h,00h], Unsigned16, read/write

The value of this entry specifies in milliseconds the time between transmission of heartbeat messages. A value of zero disables transmission of heartbeat messages by the node.

Because the entry is writeable, the value of the entry may change at any time.

As the heartbeat production is internally implemented with the lowest priority, delays in heartbeat production can be directly used to monitor the overall load and health of a device.

A.2.14 Identity (1018h)

[1018h,00h], Unsigned8, read only
Returns 3

[1018h,01h], Vendor ID, Unsigned32, read only

[1018h,02h], Product ID, Unsigned32, read only

[1018h,03h], Product Revision, Unsigned32, read only

The Identity entries are used to identify specific CANopen products and their versions. The values returned are described in more detail in section 4.1 on page 23.

A.2.15 Verify Configuration (1020h)

[1020h,00h], Unsigned8, read only

Returns 2

[1020h,01h], Configuration Date, Unsigned8, read only

[1020h,02h], Configuration Time, Unsigned8, read only

This entry allows an NMT Master or a Configuration Manager to determine, if the configuration of the device matches a known configuration.

When storing a new configuration for the node by writing to entry 1010h, a Configuration Manager first writes the current date and time to this entry along with storing the current date and time in its local OD.

Whenever any new values are written to the Object Dictionary of the node, it must set the current date and time stored in this entry to zero to indicate that the configuration has changed.

The next time the node is started, the configuration manager can read this entry and compare the date and time with the date and time stored locally in the manager. If the times match then the manager knows the current configuration of the node without having to read any further Object Dictionary entries.

The date value is the number of whole days since January 1st 1984. The time value is the number of milliseconds since midnight.

A.2.16 Receive PDO Communication Parameters (1400h - 1403h)

[140xh,00h], Unsigned8, read-only
Returns 2

[140xh,01h], COB ID, Unsigned32, read/write

[140xh,02h], Transmission Type, Unsigned8, read/write

These entries describe the communication configuration of the four Receive PDOs.

Subentry 01h defines the COB ID of the PDO. The default value depends on the index of the entry, as shown in the following table.

Index	Default Value
1400h	Node ID + 00000200h
1401h	Node ID + 00000300h
1402h	Node ID + 00000400h
1403h	Node ID + 00000500h

The bits in the COB ID entry are used as follows.

Bit	Description
0-10	COB ID for PDO
11-28	Set to 0
29	Set to 0 to select 11-bit COB ID
30	Set to 0
31	Set to 1 if the PDO is currently disabled

To change a COB ID, bit 31 must first be set to 1 to disable the PDO. Once the COB ID has been changed the PDO can be re-enabled by clearing bit 31.

Subentry 02h specifies the Transmission Type of the Receive PDO. The following table lists the available Transmission Types for a Receive PDO.

Transmission Type	Description
0-240	The Receive PDO is synchronous. The data in the PDO is processed on reception of the next SYNC Object. The actual value of the Transmission Type is not relevant.
241-253	Not used
254	Not used
255	The Transmission Type of the Receive PDO is device profile specific. The Receive PDO is asynchronous. As soon as the PDO arrives the data is processed by the node.

A.2.17 Receive PDO Mapping Parameters (1600h – 1603h)

[160xh,00h], unsigned8, read/write

[160xh,0yh], Mapping Entry, Unsigned32, read/write

This entry defines which process data is stored in a single PDO, along with the position of the process data in the eight data bytes of the PDO. A total of 4 indexes (1A00h to 1A03h) with each up to 8 Subindexes are implemented.

Each Receive PDO supported by the node must have a corresponding Receive PDO Mapping parameter entry implemented. The entry at 1600h is for the first Receive PDO whose communication parameters are defined at 1400h. The entry at 1601h is for the second Receive PDO whose communication parameters are defined at 1401h, etc.

A PDO may have 1 to 8 process data variables mapped to it, with each variable having the length of 8, 16, or 32 bits, however the total size of all the process data mapped to a single PDO may not exceed 64 bits (eight bytes). Each subentry defines a process data variable. Therefore subentry 00h holds the total number of process data variables mapped to the PDO.

The value of each subentry defines the process data variable to be mapped and the size of the process data variable in bits. The

process data variable is defined by specifying the Object Dictionary location where the data is stored. The value is constructed as follows:

Bit	Description
0-7	Data length in bits (08h, 10h or 20h)
8-15	Subindex of OD entry mapped
16-31	Index of OD entry mapped

For example, if a 16-bit process data variable was stored in the Object Dictionary at index 6001h, Subindex 04h, then it can be mapped into a PDO using the value 60010410h.

It is possible to create gaps in the mapping by using dummy entries. A dummy entry is created by mapping one of the data types located at indexes 0005h – 0007h into the PDO. For example, to create a gap of 16 bits in the PDO, the Unsigned16 data type must be defined in a subentry. This is achieved using the Unsigned16 Object Dictionary location of index 0006h Subindex 00h, giving a value for the subentry of 00060010h.

In order to change the current mapping of a PDO, the PDO must first be disabled by writing zero to subentry 00h. Once the new values for the subentries have been written, subentry 00h can be written with the number of process data variables mapped to the PDO. Attempting to write a non-zero value to subentry 00h will cause the node to check and ensure the entire mapping is valid. For example, the total number of bits mapped to the PDO does not exceed 64, each mapped process data variable exists in the Object Dictionary and can be mapped to a PDO. If the mapping is not valid, then the node will return an SDO Abort message in response to attempting to set subentry 00h to a non-zero value.

Each time a mapping entry is written, the node will check and ensure that the process data exists and can be mapped. If it does not exist or cannot be mapped then an SDO Abort message will be returned.

A.2.18 Transmit PDO Communication Parameters (1800h - 1803h)

[140xh,00h], Unsigned8, read-only

Returns 5

[140xh,01h], COB ID, Unsigned32, read/write

[140xh,02h], Transmission Type, Unsigned8, read/write

[140xh,03h], Inhibit Time, Unsigned16, read/write

[140xh,05h], Event Time, Unsigned16, read/write

These entries describe the communication configuration of the Transmit PDOs.

Subentry 01h defines the COB ID of the PDO. The default value depends on the index of the entry, as shown in the following table.

Index	Default Value
1400h	Node ID + 00000180h
1401h	Node ID + 00000280h
1402h	Node ID + 00000380h
1403h	Node ID + 00000480h

The COB ID entry also indicates if the PDO is used or not, size of the identifier and whether remote transmit requests are allowed for the PDO:

Bit	Description
0-10	COB ID for PDO
11-28	Set to 0
29	Set to 0 to select 11-bit COB ID
30	Set to 0 if remote transmit requests are allowed for the PDO, else 1
31	Set to 1 if the PDO is currently disabled

To change a COB ID, bit 31 must first be set to 1 to disable the PDO. Once the COB ID has been changed the PDO can be re-enabled by clearing bit 31.

Subentry 02h specifies the Transmission Type of the Transmit PDO. The following table lists the available Transmission Types for a Transmit PDO.

Transmission Type	Description
0	The Transmit PDO is synchronous. It is transmitted within the Synchronous Window Length after a SYNC Object. Additional details of the PDO transmission are given in the device profile.
1-240	The Transmit PDO is synchronous. It is transmitted after every n^{th} SYNC Object within the Synchronous Window Length, where n is the Transmission Type. For example, when using Transmission Type 34, the PDO is transmitted after every 34th SYNC Object.
241-251	Not used
252	The data for the PDO is updated on reception of a SYNC Object, but the PDO is not transmitted. The PDO is only transmitted on reception of a Remote Transmit Request.
253	The data for the PDO is updated and the PDO is transmitted on reception of a remote transmission request.
254	Not used
255	The Transmission Type of the Receive PDO is device profile specific. The Receive PDO is asynchronous. As soon as the PDO arrives the data is processed by the node.

Subindex 03h defines the inhibit time for the PDO. The inhibit time specifies the minimum time between transmissions of the PDO. Once the PDO is transmitted, any additional transmissions of the PDO will not take place during the inhibit time.

The inhibit time is a multiple of $100\mu\text{s}$.



Note: The base timer used in the CANopen firmware only has a resolution of 1 ms, therefore the inhibit time is always rounded up to the next millisecond.

The inhibit time is measured from the time when the node first attempts to send the PDO. If the PDO is blocked from being sent because of higher priority messages on the bus, then the delay before the PDO is actually transmitted is included in the inhibit time.

Therefore the inhibit time must be greater than the worst case transmission time of the PDO.

The inhibit time may not be changed while the PDO is being used by the node. To change the inhibit time the PDO must first be disabled by setting bit 31 of subentry 01h.

Subentry 04h is not used.

Subentry 05h defines the event time for a Transmit PDO. A value of zero disables the event timer.

If the event timer is used, then the PDO is periodically transmitted. The value of the event timer entry is the number of milliseconds between transmissions. Each time the PDO is transmitted as a result of the event timer expiring, the event timer is reset.

A.2.19 Transmit PDO Mapping Parameters (1A00h – 1A03h)

[1A0xh,00h], Unsigned8, read write

[1A0xh,0yh], Mapping Entry, Unsigned32, read/write

This entry defines which process data is stored in a single PDO, along with the position of the process data in the eight data bytes of the PDO. A total of 4 indexes (1A00h to 1A03h) with each up to 8 Subindexes are implemented.

Each Transmit PDO supported by the node must have a corresponding Transmit PDO Mapping parameter entry implemented. The entry at 1A00h is for the first Transmit PDO whose communication parameters are defined at 1800h. The entry at 1A01h is for the second Receive PDO whose communication parameters are defined at 1801h, etc.

A PDO may have 1 to 8 process data variables mapped to it, with each variable having the length of 8, 16, or 32 bits, however the total size of all the process data mapped to a single PDO may not

exceed 64 bits (eight bytes). Each subentry defines a process data variable. Therefore subentry 00h holds the total number of process data variables mapped to the PDO.

The value of each subentry defines the process data variable to be mapped and the size of the process data variable in bits. The process data variable is defined by specifying the Object Dictionary location where the data is stored. The value is constructed as follows:

Bit	Description
0-7	Data length in bits (08h, 10h or 20h)
8-15	Subindex of OD entry mapped
16-31	Index of OD entry mapped

For example, if a 16-bit process data variable was stored in the Object Dictionary at index 6001h, Subindex 04h, then it can be mapped into a PDO using the value 60010410h.

In order to change the current mapping of a PDO, the PDO must first be disabled by writing zero to subentry 00h. Once the new values for the subentries have been written, subentry 00h can be written with the number of process data variables mapped to the PDO. Attempting to write a non-zero value to subentry 00h will cause the node to check and ensure the entire mapping is valid. For example, the total number of bits mapped to the PDO does not exceed 64, each mapped process data variable exists in the Object Dictionary and can be mapped to a PDO. If the mapping is not valid, then the node will return an SDO Abort message in response to attempting to set subentry 00h to a non-zero value.

Each time a mapping entry is written, the node will check and ensure that the process data exists and can be mapped. If it does not exist or cannot be mapped then an SDO Abort message will be returned.

A.2.20 Program Download (1F50h), Node ID, and BPS

[1F50h,00h], Unsigned8, read only

Returns 3

[1F50h,03h], Hardware Settings, Unsigned32, read/write

This entry allows changing the CAN bit rate and the CANopen Node ID used by the device. After writing new settings to this OD entry, the device must be reset.

To change the CANopen Node ID, the ASCII string "Nlxx" must be written to this OD entry, where "xx" is replaced with two hexadecimal digits selecting a Node ID in the range from 1 to 127.

To change the CAN bit rate, the ASCII string "BPSx" must be written to this OD entry, where "x" is replaced with a digit in the range from 3 to 8 representing the following CAN bit rates (values are LSS compatible):

Value	Bit Rate
0	1 Mbit/s
1	800 kbit/s
2	500 kbit/s
3	250 kbit/s
4	125 kbit/s
6	50 kbit/s

Other bit rates are not supported.

A.3 Device Profile Entries

A.3.1 Read Digital Inputs (6000h)

[6000h,00h], Unsigned8, read only
If digital inputs are available, returns 1

[6000h,01h], Unsigned8, read only, can be mapped to
TPDO

For devices with digital inputs, the digital input data can be read from **[6000h,01h]**. If less than 8 input bits are implemented, the most significant bits of this entry are unused.

A.3.2 Polarity Digital Input (6002h)

[6002h,00h], Unsigned8, read only
If digital inputs are available, returns 1

[6002h,01h], Unsigned8, read/write

For each input bit available in **[6000h,01h]** this entry applies a polarity change. If a bit in this entry is set, the corresponding bit in **[6000h,01h]** is inverted. If less than 8 input bits are implemented, the most significant bits of this entry are unused. The bit is inverted when reading from the input pin, before storing into the OD.

A.3.3 Write Digital Outputs (6200h)

[6200h,00h], Unsigned8, read/write
If digital outputs are available, returns 1

[6200h,01h], Unsigned8, read only, can be mapped to
RPDO

For devices with digital outputs, the digital output data is written to **[6200h,01h]**. If less than 8 output bits are implemented, the most significant bits of this entry are unused.

A.3.4 Polarity Digital Output (6202h)

[6202h,00h], Unsigned8, read only
If digital outputs are available, returns 1

[6202h,01h], Unsigned8, read/write

For each output bit available in [6200h,01h] this entry applies a polarity change. If a bit in this entry is set, the corresponding bit from [6200h,01h] is inverted before it gets applied to the output. If less than 8 output bits are implemented, the most significant bits of this entry are unused.

A.3.5 Digital Output Error Mode (6206h)

[6206h,00h], Unsigned8, read only
If digital outputs are available, returns 1

[6206h,01h], Unsigned8, read/write

For each output bit available in [6200h,01h] this entry specifies if the error output [6207h,01h] should be applied to the output when the node goes into the pre-operational or stopped state. If a corresponding bit is set, then the error output as defined in [6207h,01h] is applied. If less than 8 output bits are implemented, the most significant bits of this entry are unused.

A.3.6 Digital Output Error Value (6207h)

[6207h,00h], Unsigned8, read only
If digital outputs are available, returns 1

[6207h,01h], Unsigned8, read/write

For each output bit available in [6200h,01h] this entry specifies the error output that should be applied when the node goes into the pre-operational or stopped state. The error output only gets applied, if the corresponding bit in [6206h,01] enables this functionality. If less than 8 output bits are implemented, the most significant bits of this entry are unused.

A.3.7 Read Analog Inputs (6401h)

[6401h,00h], Unsigned8, read only

If analog inputs are available, returns the number of input channels

[6401h,0xh], Integer16, read only, can be mapped to TPDO

For devices with analog inputs, the analog input data can be read from [6401h,01h-08h], depending on number of input channels available. The value range for each analog input channel is 0 to 1023d/03FFh (-32,768d/0h to +32,767d/FFFFh for Analog 2 motherboard).

A.3.8 Write Analog Outputs (6411h)

[6411h,00h], Unsigned8, read only

If analog outputs are available, returns the number of output channels

[6411h,0xh], Integer16, read/write, can be mapped to RPDO

For devices with analog outputs, the analog output data can be written to [6411h,01h-04h], depending on number of output channels available. The value range for each analog output channel is 0 to 255d/0FFh (0 to 4095d/FFFh for Analog 2 motherboard).

A.3.9 Analog Input Interrupt Enable (6423h)

[6423h,00h], Boolean, read/write

If this bit is set, the value difference (delta) detection for analog inputs is enabled. If the device is configured to transmit PDOs on a change-of-state (COS) detection, then this functionality ensures that a COS detection is only made if the analog input data changed at least by the difference value specified in [6426h]. Changes that are smaller than the value difference are ignored.

A.3.10 Analog Input Value Difference (6426h)

[6426h,00h], Unsigned8, read only

If analog inputs are available, returns the number of input channels

[6426h,0xh], Unsigned32, read/write

For each analog input channel there is one subentry specifying the minimum value difference (delta) used for the channel. Value difference detection is only enabled if the bit **[6423h,00h]** is set and the TPDO containing the analog data uses change-of-state detection (Transmission Type 255).

A.3.11 Analog Output Error Mode (6443h)

[6443h,00h], Unsigned8, read only

If analog outputs are available, returns the number of output channels

[6443h,0xh], Unsigned8, read/write

There is one subentry for each output channel **[6411h,0xh]** available specifying if the corresponding error output **[6444h,0xh]** should be applied to the output when the node goes into the pre-operational or stopped state. When a subentry is set to one, the corresponding error output is enabled. When set to zero, it is disabled.

A.3.12 Analog Output Error Value (6444h)

[6444h,00h], Unsigned8, read only

If analog outputs are available, returns the number of output channels

[6444h,0xh], Integer32, read/write

There is one subentry for each output channel **[6411h,0xh]** available specifying the error output that should be applied when the node goes into the pre-operational or stopped state. The error output only gets applied, if the corresponding subentry **[6443h,0xh]** enables this functionality.